Journal of Discrete Algorithms 16 (2012) 170–186

Contents lists available at SciVerse ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Strengthening hash families and compressive sensing ☆

Charles J. Colbourn [a,*], Daniel Horsley [b], Violet R. Syrotiuk [a]

[a] Computing, Informatics, and Decision Systems Engineering, Arizona State University, P.O. Box 878809, Tempe, AZ 85287, USA
[b] School of Mathematical Sciences, Monash University, Victoria 3800, Australia

ARTICLE INFO

Article history:
Available online 3 April 2012

Keywords:
Compressive sensing
Hash family
Column replacement
Stein–Lovász–Johnson theorem
Lovász local lemma

ABSTRACT

The deterministic construction of measurement matrices for compressive sensing is a challenging problem, for which a number of combinatorial techniques have been developed. One of them employs a widely used column replacement technique based on hash families. It is effective at producing larger measurement matrices from smaller ones, but it can only preserve the strength (level of sparsity supported), not increase it. Column replacement is extended here to produce measurement matrices with larger strength from ingredient arrays with smaller strength. To do this, a new type of hash family, called a strengthening hash family, is introduced. Using these hash families, column replacement is shown to increase strength under two standard notions of recoverability. Then techniques to construct strengthening hash families, both probabilistically and deterministically, are developed. Using a variant of the Stein–Lovász–Johnson theorem, a deterministic, polynomial time algorithm for constructing a strengthening hash family of fixed strength is derived.

1. Introduction

Sparse measurement, testing, and location problems abound. Locating a small number of defectives in a large population (combinatorial group testing [31]), learning a function with few relevant attributes [27], locating a fault in a large component-based software system (software interaction testing [49]), and recovering a sparse signal from a 'small' set of samples (compressive sensing [2]) have much in common. In each case, many factors (coordinates, attributes) can each be set to one of a set of values (levels) to form a test (sample). Selected tests can then be run to determine an outcome, or measurement. From the resulting measurements, we are to find significant factors, and in some cases significant interactions among the factors. Of course, the details in each case mentioned vary substantially, but the basic framework is the same.

In each of the cases mentioned, we are to construct arrays in which each row is a test; arrays with the fewest rows for the specified number of columns are always preferred. When the interactions to be found involve few of the factors (say at most t), the tests must together only reveal interactions of size t or less; this is the *strength* of the array required. Now when the strength is fixed, in each of the problems mentioned, elementary probabilistic arguments show that the number of tests needed grows as the logarithm of the number of factors. This underpins a substantial focus on sparsity in testing and measurement.

While probabilistic techniques provide the correct asymptotics for each problem mentioned, actual problems require the explicit construction of arrays of tests. Hence explicit, deterministic constructions are of interest. This has been best examined in the software interaction testing domain. There, while probabilistic techniques yield the best results asymptotically,

☆ A preliminary version of some of this paper appears in Colbourn (2011) [21].

* Corresponding author.
E-mail addresses: colbourn@asu.edu (C.J. Colbourn), danhorsley@gmail.com (D. Horsley), syrotiuk@asu.edu (V.R. Syrotiuk).

1570-8667/$ – see front matter
http://dx.doi.org/10.1016/j.jda.2012.04.004

a combination of computational methods, direct constructions using finite fields and finite groups, and recursive constructions, yields substantially smaller arrays when the number of factors is in the hundreds or thousands [20]. Computation in that case is effective for tens or hundreds of factors, but the workhorses in producing the best known array sizes are recursive constructions that make larger arrays from smaller ones.

A major recursive construction uses a pattern matrix P to select columns from a small array A (or more generally a set $\{A_i\}$ of small arrays) to construct a larger array. For this column replacement technique (developed in more detail in Section 3) to produce an array of tests that treats more factors, at the same strength, one needs to ensure that the pattern matrix selects suitable sets of columns. Thus the pattern array must exhibit a certain combinatorial structure; fortunately, the required structure is already well studied in the guise of hash families. Hash families and their variants have been explored for numerous applications (see [20,26,52], for example), among them applications in hashing [45].

The application of column replacement is pervasive, being used in software interaction testing [20], hash families [5], and even compressive sensing [22]. Presently the most serious limitation of column replacement techniques is that, from ingredient arrays of strength t, the larger array produced is also of strength t. Thus, although the number of factors may increase dramatically, the strength remains unchanged. Put another way, to employ column replacement to make an array of large strength, one needs ingredient arrays of large strength. This is one main reason why arrays for software interaction testing have been studied primarily for strengths of at most 6.

The main contribution of this paper is the extension of column replacement to make arrays of larger strength from arrays with smaller strength. To do this, we extend the definitions of hash families to incorporate a *strengthening* requirement, which – as one should expect – is precisely what is needed for column replacement to increase the strength. We select compressive sensing to illustrate strengthening in the construction of measurement matrices, generalizing the previous column replacement technique; similar illustrations can be found in the other problem domains. We establish the required results on recoverability using two standard recovery schemes, and examine the effect of column replacement on the so-called 'restricted isometry property' (RIP) parameters.

For the method to be more than a curiosity, we must construct strengthening hash families. We establish that two well known probabilistic techniques, the Lovász local lemma and the Stein–Lovász–Johnson method, produce both effective asymptotic results and probabilistic algorithms. A careful analysis of the Stein–Lovász–Johnson approach enables us to develop an algorithm meeting the logarithmic bound on the number of tests whose running time is polynomial in the number of factors, when the strength is fixed. A small set of computational results demonstrate that the method developed is effective.

The remainder of the paper is organized as follows. Section 2 formally defines a general notion of hash families, as well as several variants including the extension to strengthening hash families. Section 3 develops the use of such hash families in the construction of measurement matrices for compressive sensing. In particular, the basics of compressive sensing for ℓ_0- and ℓ_1-recovery are developed in Section 3.1; column replacement for measurement matrices using separating and distributing hash families is introduced in Section 3.2; column replacement is extended to strengthening hash families in Section 3.3; and the impact of column replacement on the RIP parameters is examined in Section 3.4. Because strengthening hash families must be constructed in order to make these column replacement strategies effective, the remainder of the paper treats their construction. The Lovász local lemma is used in Section 4 to develop asymptotic existence results, and both deterministic and probabilistic methods for their construction. Section 5 instead applies the Stein–Lovász–Johnson method to develop asymptotic existence results and a probabilistic algorithm. Section 6 then establishes that, for a wide variety of types of hash families, including strengthening hash families of fixed strength, the Stein–Lovász–Johnson method can be implemented to run in polynomial time in the number of columns. Section 7 reports a limited set of computational results on strengthening hash families from an implementation of this method. Finally, Section 8 draws relevant conclusions.

2. Strengthening hash families

Let N, k, and v be positive integers. A *hash family* $\mathsf{HF}(N; k, v)$, $A = (a_{ij})$, is an $N \times k$ array, in which each cell contains one symbol from a set Σ of v symbols. We generalize this standard definition by replacing v by $\mathbf{v} = (v_1, \ldots, v_N)$, a tuple of positive integers. A *hash family* $\mathsf{HF}(N; k, \mathbf{v})$, $A = (a_{ij})$, is an $N \times k$ array, in which each cell contains one symbol, and for $1 \leqslant \rho \leqslant N$, $|\{a_{\rho j}\colon\ 1 \leqslant j \leqslant k\}| \leqslant v_\rho$. When $v_1 = \cdots = v_N = v$ (i.e., the array is *homogeneous*), we recover the standard definition.

Nevertheless, we permit *heterogeneity*, i.e., we allow that v_i and v_j are not necessarily equal when $i \neq j$, because it is useful in certain applications of hash families in column replacement techniques [20,23,24], one of which we develop in Section 3.

Many variants of hash families have been studied. An $\mathsf{HF}(N; k, \mathbf{v})$, $A = (a_{ij})$, is

***perfect* of *strength* t:** denoted $\mathsf{PHF}(N; k, \mathbf{v}, t)$, when for every set C of at most t columns, there exists a row ρ for which $|\{a_{\rho c}\colon\ c \in C\}| = |C|$ (see [1,52], for example);

$\{w_1, w_2, \ldots, w_s\}$-***separating*:** denoted $\mathsf{SHF}(N; k, \mathbf{v}, \{w_1, w_2, \ldots, w_s\})$, if whenever C is a set of $\sum_{i=1}^{s} w_i$ columns and $C_1, C_2, \ldots, C_s$ is a partition of C with $|C_i| = w_i$ for $1 \leqslant i \leqslant s$, there exists a row ρ for which $a_{\rho x} \neq a_{\rho y}$ whenever $x \in C_i$, $y \in C_j$ and $i \neq j$ (see [6,53], for example).

								↓	↓	↓		
	0	1	2	2	1	2	2	0	1	1	0	0
	0	2	1	0	2	2	2	1	0	1	2	1
	1	0	0	2	2	2	1	1	2	1	0	2
	2	0	1	1	2	0	2	0	1	1	2	1
	2	0	2	1	2	1	0	2	2	1	1	0
→	2	0	1	2	1	1	2	[2]	[0]	[1]	2	1

Fig. 1. A homogeneous perfect hash family PHF(6; 12, 3, 3).

										↓				↓	↓
	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
→	1	2	3	4	2	1	4	3	3	4	[1]	2	4	3	[2]	[1]

Fig. 2. A homogeneous {1, 2}-separating hash family SHF(3; 16, 4, (1, 2)).

					↓	↓				↓		↓	↓
	6	7	8	3	4	0	2	2	3	0	5	1	1
	3	1	1	7	2	6	8	4	3	0	2	0	5
	8	5	1	4	2	3	2	6	7	0	1	3	0
	0	2	0	2	2	0	0	1	1	1	1	2	0
	0	0	2	1	1	1	2	0	0	2	2	0	1
→	1	1	2	2	[2]	[0]	1	0	0	[2]	1	[0]	[0]
	1	0	1	2	0	0	2	0	0	1	2	2	1
	1	1	0	1	0	3	2	0	2	0	1	0	2
	0	0	3	0	1	0	0	2	4	0	0	1	0
	0	⋆	⋆	⋆	⋆	1	⋆	⋆	1	⋆	⋆	0	1

Fig. 3. A heterogeneous DHF(10; 13, **v**, 5, 2) with $\mathbf{v} = (9^3 3^4 4^1 5^1 2^1)$.

$\mathcal{W}$-***separating*****:** denoted SHF$(N; k, \mathbf{v}, \mathcal{W})$ for $\mathcal{W}$ a set of multisets of nonnegative integers of the form $\{w_1, w_2, \ldots, w_s\}$, if it is $\{w_1, w_2, \ldots, w_s\}$-separating for each $\{w_1, w_2, \ldots, w_s\} \in \mathcal{W}$.

(t, s)-***distributing*****:** denoted DHF$(N; k, \mathbf{v}, t, s)$, if it is $\mathcal{W}$-separating with $\mathcal{W}$ containing every multiset $\{w_1, \ldots, w_s\}$ of nonnegative integers with $\sum_{i=1}^{s} w_i = t$.

An example of a homogeneous perfect hash family PHF(6; 12, 3, 3) is given in Fig. 1. It is a 6×12 array on the three symbols $\{0, 1, 2\}$ in which in every 6×3 subarray, at least one row consists of distinct symbols. For the 6×3 subarray involving columns 8, 9, and 10, only the last row consists of distinct symbols.

Fig. 2 gives an example of a homogeneous $\{1, 2\}$-separating hash family SHF(3; 16, 4, {1, 2}). It is a 3×16 array on the four symbols $\{1, 2, 3, 4\}$. For the 3×3 subarray consisting of columns 11, 15, and 16, only the last row accomplishes the specific separation of columns $\{11, 16\}$ from column $\{15\}$.

Fig. 3 gives an example of a heterogeneous DHF(10; 13, **v**, 5, 2) with $\mathbf{v} = (9, 9, 9, 3, 3, 3, 3, 4, 5, 2)$. We abbreviate repetition in such a tuple by using an exponential notation that indicates the repetition in the exponent $\mathbf{v} = (9^3 3^4 4^1 5^1 2^1)$. The cells with the '⋆' are 'flexible' positions, i.e., they can take on any symbol. The DHF is a 10×13 array on the nine symbols $\{0, 1, \ldots, 8\}$. Because $t = 5$ and $s = 2$ we must achieve both $\{1, 4\}$- and $\{2, 3\}$-separations. For the 10×5 subarray consisting of columns five, six, ten, twelve, and thirteen, only row six separates columns $\{5, 10\}$ from columns $\{6, 12, 13\}$; the ⋆ in the last row could be filled to effect the separation again, but there is no need.

The definition of $\mathcal{W}$-separating encompasses perfect, $\{w_1, w_2, \ldots, w_s\}$-separating, and (t, s)-distributing hash families, so we can treat this general situation. On occasion, we wish to further restrict the choice of rows to provide the desired separation. The choice can be defined by a logical predicate.

Let $\mathbf{d} = (d_1, \ldots, d_N)$ be a tuple of positive integers, and let τ be a positive integer. Let $\mathcal{W} = \{W_1, \ldots, W_r\}$, where for $1 \leqslant i \leqslant r$, $W_i = \{w_{i1}, \ldots, w_{is_i}\}$ is a multiset of nonnegative integers, and $\sigma_i = \sum_{j=1}^{s_i} w_{ij}$. Finally, let $\mathbf{\Pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ be a tuple of predicates on multisets of τ values that evaluate to **true** or **false**.

An SHF$(N; k, \mathbf{v}, \mathcal{W})$, $A = (a_{ij})$, is

$(\mathbf{d}, \tau)$-$\mathbf{\Pi}$: if whenever
$1 \leqslant i \leqslant r$,
C is a set of σ_i columns,
$C_1, C_2, \ldots, C_{s_i}$ is a partition of C with $|C_j| = w_{ij}$ for $1 \leqslant j \leqslant s_i$, and
T is a set of τ columns with $|C \cap T| = \min(\sigma_i, \tau)$,

there exists a row ρ for which
$a_{\rho x} \neq a_{\rho y}$ whenever $x \in C_e$, $y \in C_f$ and $e \neq f$ *and*
the predicate π_ρ evaluates to **true** on multiset $\{a_{\rho x}\colon\ x \in T\}$.

	↓	↓				↓	↓				↓		
	4	0	2	1	3	3	0	0	1	4	2	2	1
	0	0	1	1	2	3	1	3	2	4	2	0	4
	0	2	4	1	1	2	0	1	2	3	0	3	4
	2	4	1	0	3	0	3	1	1	4	2	0	2
	2	1	2	2	4	0	0	4	0	1	1	3	3
	3	4	0	1	0	3	2	4	2	1	1	2	0
	0	0	1	0	0	2	2	0	0	1	3	0	0
→	0	1	0	1	1	2	0	2	0	0	2	1	2
	1	0	2	0	0	2	1	1	0	2	0	2	1
	0	1	2	2	1	2	1	0	0	0	1	0	2
	0	2	2	2	1	2	0	0	1	1	0	0	1
	0	1	1	0	1	2	1	0	0	2	0	2	2
	2	1	0	1	2	0	1	2	0	2	0	0	1
	2	1	2	0	2	2	0	0	1	0	0	1	1
	0	0	0	1	0	1	1	2	0	1	2	2	2
	1	2	0	1	1	1	2	2	0	0	0	2	0
	1	0	2	1	1	0	0	2	0	0	2	1	0
	2	2	0	0	1	2	1	0	0	1	2	1	2
	0	0	2	1	1	⋆	1	2	0	2	2	1	1

Fig. 4. A heterogeneous **d**-strengthening DHF(19; 13, **v**, 5, 2) with $\mathbf{v} = (5^6 4^1 3^{12})$, $\mathbf{d} = (4^6 3^{13})$.

There are at least two natural choices of predicate. An $\mathsf{SHF}(N; k, \mathbf{v}, \mathcal{W})$, $A = (a_{ij})$, is

(**d**, τ)-***scattering*:** The predicate $\boldsymbol{\pi_\rho}$ evaluates to **true** if the multiset $\{a_{\rho x}\colon\ x \in T\}$ contains no symbol more than d_ρ times, and **false** otherwise.

(**d**, τ)-***strengthening*:** The predicate $\boldsymbol{\pi_\rho}$ evaluates to **true** if the multiset $\{a_{\rho x}\colon\ x \in T\}$ contains no more than d_ρ different symbols, and **false** otherwise.

When $\tau = \max\{\sigma_i\colon\ 1 \leqslant i \leqslant r\}$, we omit τ from the notation and write **d**-scattering or **d**-strengthening. When $d_1 = \cdots = d_N = d$, we write d in place of **d**.

Fig. 4 depicts a heterogeneous **d**-strengthening DHF(19; 13, **v**, 5, 2) with $\mathbf{v} = (5^6 4^1 3^{12})$ and $\mathbf{d} = (4^6 3^{13})$. This is equivalent to a **d**-strengthening $\mathsf{SHF}(19; 13, \mathbf{v}, \{\{1, 4\}, \{2, 3\}\})$. Consider the separation of columns $\{1, 7\}$ from columns $\{2, 6, 11\}$. Row 8 accomplishes the required separation, and it satisfies predicate π_8 because it uses $d_8 = 3$ symbols in these five columns. However, considering columns $\{1, 2, 3, 4, 5\}$, the first row separates $\{1, 2, 3\}$ from $\{4, 5\}$, but does not satisfy predicate π_1 because it uses 5 symbols to separate and π_1 only permits the use of $d_1 = 4$ symbols. So this separation is required in another row (in this case, row 3 is sufficient).

O'Brien [50] proposed the scattering requirement for a construction for a broadcast encryption scheme. In this paper, we explore the need for strengthening hash families, by examining their application in the construction of measurement matrices for compressive sensing.

Our concern throughout is to find hash families with as few rows as possible, given the other parameters. How many rows are necessary? While exact determinations are known in very few cases, a naive lower bound yields the 'correct' asymptotic growth rate. To avoid trivialities, a set $\mathcal{W}$ is *binding* if there exists a $\{w_1, \ldots, w_s\} \in \mathcal{W}$ that contains at least two positive entries. (When $\mathcal{W}$ is not binding, an $\mathsf{SHF}(N; k, \mathbf{v}, \mathcal{W})$ may have $N = 0$.) When $\mathcal{W}$ is binding, an $\mathsf{SHF}(N; k, \mathbf{v}, \mathcal{W})$ cannot have two identical columns, and hence $k \leqslant \prod_{i=1}^{N} v_i$. Treating the homogeneous case with v symbols, the number of rows must be at least $\lceil \log_v k \rceil$. This yields a *logarithmic lower bound*. More precise bounds are known in many cases (for example, [3,6]), but all are $\Omega(\log k)$ bounds on N. When d, v, and $\mathcal{W}$ are fixed; $v \geqslant d$; and $d \geqslant \max\{s\colon\ \{w_1, \ldots, w_s\} \in \mathcal{W}\}$, Theorem 5.4 given later establishes that there is an absolute constant c so that a d-strengthening $\mathsf{SHF}(N; k, \mathbf{v}, \mathcal{W})$ exists having $N \leqslant \lfloor c \log k \rfloor$. Hence the lower and upper bounds differ by a constant factor.

3. Column replacement for compressive sensing

3.1. Compressive sensing

Nyquist's sampling theorem states that a bandlimited analog signal can be perfectly reconstructed if the sampling rate is at least twice the highest frequency of the original signal. However, when the signal is sparse, far fewer samples than the Nyquist rate are required. *Compressive sensing* [9,2] (also called *compressive sampling*) is a technique that captures and represents such compressible signals, in a sense combining compression and sampling into a single step.

We begin by describing a specific framework. An *admissible signal* of *dimension* n is a vector in $\mathbb{R}^n$ which is known *a priori* to be taken from a given set $\Phi \subseteq \mathbb{R}^n$. A *measurement matrix* A is a matrix from $\mathbb{R}^{m \times n}$. *Sampling* a signal $\mathbf{x} \in \mathbb{R}^n$ is computing the product $A\mathbf{x} = \mathbf{b}$. Once sampled, *recovery* involves determining the unique signal $\mathbf{x} \in \Phi$ that satisfies $A\mathbf{x} = \mathbf{b}$ using only A and $\mathbf{b}$. If $\Phi = \mathbb{R}^n$, recovery can be accomplished only if A has rank n, and hence $m \geqslant n$. However for more restrictive admissible sets Φ, recovery can sometimes be accomplished when $m < n$. Given a measurement matrix A, define an equivalence relation $\equiv_A$ so that for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have $\mathbf{x} \equiv_A \mathbf{y}$ if and only if $A\mathbf{x} = A\mathbf{y}$. If, for every equivalence class P

under $\equiv_A$, the set $P \cap \Phi$ contains at most one signal, then recovery is possible in principle. Because $A\mathbf{x} = A\mathbf{y}$ ensures that $A(\mathbf{x}-\mathbf{y}) = \mathbf{0}$, this can be stated more simply. The *null space* of A, $N(A)$, is the set $\{\mathbf{x} \in \mathbb{R}^n\colon\ A\mathbf{x} = \mathbf{0}\}$. An equivalence class P of $\equiv_A$ can be represented as $\{\mathbf{x}+\mathbf{y}\colon\ \mathbf{y} \in N(A)\}$ for any $\mathbf{x} \in P$. Hence recoverability is equivalent to requiring that, for every signal $\mathbf{x} \in \Phi$, there is no $\mathbf{y} \in N(A) \setminus \{\mathbf{0}\}$ with $\mathbf{x}+\mathbf{y} \in \Phi$.

To apply these observations, a reasonable *a priori* restriction on the signals to be sampled is identified, suitable measurement matrices with $m \ll n$ are formed, and a reasonably efficient computational strategy for recovering the signal is provided. A signal is *t-sparse* if at most t of its n coordinates are nonzero. The recovery of t-sparse signals is the province of *compressive sensing*. An admissible set of signals Φ has *sparsity* t when every signal in Φ is t-sparse. An admissible set of signals Φ is *t-sparsifiable* if there is a full rank matrix $B \in \mathbb{R}^{n\times n}$ for which $\{B\mathbf{x}\colon\ \mathbf{x} \in \Phi\}$ has sparsity t. Sparse and sparsifiable signals arise in numerous applications in sensing, imaging, and communications [2]. We assume throughout that when the signals are sparsifiable, a change of basis B is applied so that the admissible signals have sparsity t.

A measurement matrix *has (ℓ_0, t)-recoverability* when it permits exact recovery of all t-sparse signals. A basic problem is to design measurement matrices with (ℓ_0, t)-recoverability where m is small relative to n, but in such a way that recovery can be accomplished efficiently. Suppose that matrix A has (ℓ_0, t)-recoverability. Then given A and $\mathbf{b}$, recovery of the signal $\mathbf{x}$ can be accomplished in principle by solving the ℓ_0-minimization problem $\min\{\|\mathbf{x}\|_0\colon\ A\mathbf{x} = \mathbf{b}\}$. An enumerative strategy can be employed: List the possible supports of signals from fewest nonzero entries to most. For each, reduce A and $\mathbf{x}$ to A' and $\mathbf{x}'$, respectively, by eliminating coordinates in the signal assumed to be zero. Examine the (now overdetermined) system $A'\mathbf{x}' = \mathbf{b}$. When equality holds, a solution is found; we are guaranteed to find one by considering all possible supports with at most t nonzero entries. This strategy is prohibitively time-consuming, examining as many as $\binom{n}{t}$ linear systems when the signal has sparsity t. Natarajan [47] showed that we cannot expect to find a substantially more efficient solution, because the problem is NP-hard.

Chen, Donoho, and Saunders [15,30] instead suggest considering the ℓ_1-minimization problem $\min\{\|\mathbf{x}\|_1\colon\ A\mathbf{x} = \mathbf{b}\}$, which can be solved using standard linear programming techniques. For this to be effective, it is necessary that for each t-sparse signal $\mathbf{x}$, $\mathbf{x}$ is the unique solution to $\min\{\|\mathbf{z}\|_1\colon\ A\mathbf{z} = A\mathbf{x}\}$. This property is (ℓ_1, t)-recoverability. A necessary and sufficient condition for (ℓ_1, t)-recoverability has been explored, beginning with Donoho and Huo [30] and subsequently in [54,57,32,35,36,55,39].

For $\mathbf{y} \in \mathbb{R}^n$ and $C \subset \{1, \ldots, n\}$, define $\mathbf{y}_{|C} \in \mathbb{R}^n$ to be the vector such that $(y_{|C})_\gamma = y_\gamma$ if $\gamma \in C$ and $(y_{|C})_\gamma = 0$ otherwise. A matrix A meets the *(ℓ_0, t)-null space condition* if and only if $N(A) \setminus \{\mathbf{0}\}$ contains no $(2t)$-sparse vector. A matrix A meets the *(ℓ_1, t)-null space condition* if and only if for every $\mathbf{y} \in N(A) \setminus \{\mathbf{0}\}$ and every $C \subset \{1, \ldots, n\}$ with $|C| = t$, $\|\mathbf{y}_{|C}\|_1 < \frac{1}{2}\|\mathbf{y}\|_1$.

Lemma 3.1. *(See [22], for example.) Matrix $A \in \mathbb{R}^{m\times n}$ has (ℓ_0, t)-recoverability if and only if A meets the (ℓ_0, t)-null space condition.*

Lemma 3.2. *(See [57], for example.) Matrix $A \in \mathbb{R}^{m\times n}$ has (ℓ_1, t)-recoverability if and only if A meets the (ℓ_1, t)-null space condition.*

To establish (ℓ_1, t)-recoverability (and hence also (ℓ_0, t)-recoverability), Candès and Tao [10,12] introduced the *Restricted Isometry Property* (RIP). For $A \in \mathbb{R}^{m\times n}$, the *$d$th RIP parameter* of A, $\delta_d(A)$, is the smallest δ so that, for some $R > 0$, $(1-\delta)R(\|\mathbf{x}\|_2)^2 \leqslant (\|A\mathbf{x}\|_2)^2 \leqslant (1+\delta)R(\|\mathbf{x}\|_2)^2$, for all $\mathbf{x}$ with $\|\mathbf{x}\|_0 = d$. The smaller $\delta_d(A)$ is, the better the dth RIP parameter. The RIP parameters have been extensively employed to establish (ℓ_1, t)-recoverability, particularly for randomly generated measurement matrices [11–13], but also for deterministic constructions [17,29]. Conditions on the RIP parameters have been used to provide sufficient conditions for (ℓ_1, t)-recoverability; commonly $\delta_{2t} < \sqrt{2} - 1$ is required for (ℓ_1, t)-recoverability, see [10] for example. The property of (ℓ_1, t)-recoverability in the presence of noise has also been considered. Conditions on the RIP parameters are sufficient, but in general not necessary, for recoverability.

Combinatorial methods [4,25,37,38,40,56,41] for compressive sensing have been examined using close relationships with expander graphs. We pursue a different combinatorial approach here.

3.2. Column replacement: The basics

Let $A \in \mathbb{R}^{r\times k}$, $A = (a_{ij})$, be a measurement matrix. Let $P \in \{1, \ldots, k\}^{m\times n}$, $P = (p_{ij})$, be a *pattern matrix*. The *column replacement* of A into P is a matrix $B \in \mathbb{R}^{rm\times n}$, $B = (B_{ij})$, so that $b_{(\beta-1)r+s,\gamma} = a_{s,p_{\beta\gamma}}$ for $1 \leqslant \beta \leqslant m$, $1 \leqslant \gamma \leqslant n$, and $1 \leqslant s \leqslant r$. That is, B is obtained by replacing each symbol σ in P by the column of A indexed by σ. The matrix B can be partitioned into m *bands* $B_1, \ldots B_m$, where band B_β is the $r \times n$ matrix obtained by selecting all rows indexed from $(\beta-1)r+1$ to βr. Evidently, $N(B) = \bigcap_{\beta=1}^{m} N(B_\beta)$.

Fig. 5 illustrates the idea of column replacement. In it, A is 2×3 matrix with columns indexed by $\{1, \ldots, 3\}$, P is 2×4 pattern matrix with symbols from $\{1, \ldots, 3\}$, and B is the column replacement of A into P. B is a 4×4 matrix having entries chosen from the set of entries of A. The two bands are separated by the line.

Theorem 3.3. *(See [22].) Suppose that A is an $r \times k$ measurement matrix that meets the (ℓ_0, t)-null space condition, that P is an* $\mathsf{SHF}(m; n, k, (1, t))$*, and that B is the column replacement of A into P. Then B is an $rm \times n$ measurement matrix that meets the (ℓ_0, t)-null space condition.*

As expected, the more stringent (ℓ_1, t)-recoverability condition requires more.

$$P = \begin{bmatrix} 1231 \\ 3121 \end{bmatrix} \qquad A = \begin{bmatrix} \mathtt{aba} \\ \mathtt{baa} \end{bmatrix} \qquad B = \left[\begin{array}{c} \mathtt{abaa} \\ \mathtt{baab} \\ \hline \mathtt{aaba} \\ \mathtt{abab} \end{array}\right]$$

Fig. 5. B is the column replacement of A into P.

$$P = \begin{bmatrix} 132123 \\ 111222 \end{bmatrix} \qquad A_1 = \begin{bmatrix} \mathtt{aba} \\ \mathtt{baa} \end{bmatrix} \qquad A_2 = \begin{bmatrix} \mathtt{ab} \\ \mathtt{ba} \end{bmatrix} \qquad B = \left[\begin{array}{c} \mathtt{aababa} \\ \mathtt{baabaa} \\ \hline \mathtt{aaabbb} \\ \mathtt{bbbaaa} \end{array}\right]$$

Fig. 6. B is the column replacement of A_1, A_2 into P.

Theorem 3.4. *(See [22].) Suppose that A is an $r \times k$ measurement matrix that meets the (ℓ_1, t)-null space condition, that P is a $\mathsf{DHF}(m; n, k, t+1, 2)$, and that B is the column replacement of A into P. Then B is an $rm \times n$ measurement matrix that meets the (ℓ_1, t)-null space condition.*

As discussed in [22], Theorems 3.3 and 3.4 enable one to restrict the entries of a measurement matrix to a predetermined set, rather than employing 'random' entries. In addition, the structure of the measurement matrices from column replacement exhibit a hierarchical structure that can aid in recovery, and indeed support hybrid recovery schemes. However, they do not permit the construction of measurement matrices for larger strength from ones for smaller strength. It is this shortcoming that we address next.

3.3. Strengthening with column replacement

We first extend column replacement to exploit heterogeneity in the hash family. Let $P = (p_{ij})$ be an $\mathsf{HF}(m; n, \mathbf{k})$ and, for $1 \leqslant i \leqslant m$, A_i be an $r_i \times k_i$ matrix. For each row i of P, replace the entry p_{ij} by (a copy of) the p_{ij}th column of A_i. The result is a $(\sum_{i=1}^{m} r_i) \times n$ matrix, which is the *column replacement of $A_1, A_2, \ldots, A_m$ into P*. A small example is given in Fig. 6.

Theorem 3.5. *Let $\mathbf{k} = (k_1, \ldots, k_m)$ and $\mathbf{q} = (q_1, \ldots, q_m)$ be tuples of positive integers. Let $\mathbf{d} = (2q_1, \ldots, 2q_m)$. For $1 \leqslant i \leqslant m$, let $A_i \in \mathbb{R}^{r_i \times k_i}$ be a measurement matrix that meets the (ℓ_0, q_i)-null space condition. Let P be a $(\mathbf{d}, 2t)$-strengthening $\mathsf{SHF}(m; n, \mathbf{k}, (1, t))$, and let B be the column replacement of $A_1, A_2, \ldots, A_m$ into P. Then B meets the (ℓ_0, t)-null space condition.*

Proof. Suppose to the contrary that B does not meet the (ℓ_0, t)-null space condition, and that $\mathbf{z} \in N(B) \setminus \{\mathbf{0}\}$ is a $(2t)$-sparse vector. Let $\{\rho_1, \ldots, \rho_s\} = \{\gamma\colon\ z_\gamma > 0\}$, $\{\nu_1, \ldots, \nu_{s'}\} = \{\gamma\colon\ z_\gamma < 0\}$, and $C = \{\rho_1, \ldots, \rho_s\} \cup \{\nu_1, \ldots, \nu_{s'}\}$. Then $s + s' \leqslant 2t$, and without loss of generality we can take $s \leqslant s'$ so $s \leqslant t$. Then $s' \geqslant 1$ because $\mathbf{z}$ is nonzero.

Because P is $(1, t)$-separating and $(\mathbf{d}, 2t)$-strengthening, it contains a row β such that $p_{\beta\rho_i} \neq p_{\beta\nu_1}$ for $1 \leqslant i \leqslant s$ and $|\{p_{\beta\gamma}\colon\ \gamma \in C\}| \leqslant 2q_\beta$. Because $B\mathbf{z} = \mathbf{0}$, $(B_\beta)\mathbf{z} = \mathbf{0}$. Form a vector $\mathbf{w} \in \mathbb{R}^{k_\beta}$ by setting $w_\sigma = \sum\{z_\gamma\colon\ p_{\beta\gamma} = \sigma, 1 \leqslant \gamma \leqslant n\}$ for $1 \leqslant \sigma \leqslant k_\beta$. The vector $\mathbf{w}$ can be considered as a projection of $\mathbf{z}$ onto $\mathbb{R}^{k_\beta}$ induced by the symbol pattern in row β of P. Because B is the column replacement of $A_1, A_2, \ldots, A_m$ into P, it follows from $(B_\beta)\mathbf{z} = \mathbf{0}$ that $A_\beta \mathbf{w} = \mathbf{0}$. Because $|\{p_{\beta\gamma}\colon\ \gamma \in C\}| \leqslant 2q_\beta$, $\mathbf{w}$ is $(2q_\beta)$-sparse. Finally, $\mathbf{w}$ is nonzero because $w_{p_{\beta\nu_1}} < 0$. So A_β does not meet the (ℓ_0, q_β)-null space condition, a contradiction. □

Theorem 3.6. *Let $\mathbf{k} = (k_1, \ldots, k_m)$ and $\mathbf{q} = (q_1, \ldots, q_m)$ be tuples of positive integers. For $1 \leqslant i \leqslant m$, let $A_i \in \mathbb{R}^{r_i \times k_i}$ be a measurement matrix that meets the (ℓ_1, q_i)-null space condition. Let P be a $(\mathbf{q}, t)$-strengthening $\mathsf{DHF}(m; n, \mathbf{k}, t+1, 2)$, and let B be the column replacement of $A_1, A_2, \ldots, A_m$ into P. Then B meets the (ℓ_1, t)-null space condition.*

Proof. Suppose to the contrary that B does not meet the (ℓ_1, t)-null space condition. Let $\mathbf{z} \in \mathbb{R}^n$, $C = \{\gamma_1, \ldots, \gamma_t\}$, and $\overline{C} = \{1, \ldots, n\} \setminus C$ so that $\mathbf{z} \in N(B) \setminus \{\mathbf{0}\}$ and $\|\mathbf{z}_{|C}\|_1 \geqslant \|\mathbf{z}_{|\overline{C}}\|_1$. Let χ be an element of $\overline{C}$ such that $|z_\chi| \geqslant |z_\gamma|$ for each $\gamma \in \overline{C}$. Suppose without loss of generality that $z_\chi \geqslant 0$. We shall obtain a contradiction by examining a band of B corresponding to a row of P that, using few symbols, separates those indices in $C \cup \{\chi\}$ for which $\mathbf{z}$ has nonnegative entries from those indices in $C \cup \{\chi\}$ for which $\mathbf{z}$ has negative entries.

Because P is $(t+1, 2)$-distributing and $(\mathbf{q}, t)$-strengthening, there is a row β of P such that

$$\left\{p_{\beta\gamma}\colon\ \gamma \in C \cup \{\chi\},\ z_\gamma \geqslant 0\right\} \cap \left\{p_{\beta\gamma}\colon\ \gamma \in C \cup \{\chi\},\ z_\gamma < 0\right\} = \emptyset$$

and $|\{p_{\beta\gamma}\colon\ \gamma \in C\}| \leqslant q_\beta$.

Because $B\mathbf{z} = \mathbf{0}$, $(B_\beta)\mathbf{z} = \mathbf{0}$. Form a vector $\mathbf{w} \in \mathbb{R}^{k_\beta}$ by setting $w_\sigma = \sum\{z_\gamma\colon\ p_{\beta\gamma} = \sigma, 1 \leqslant \gamma \leqslant n\}$ for $1 \leqslant \sigma \leqslant k_\beta$. The vector $\mathbf{w}$ can be considered as a projection of $\mathbf{z}$ onto $\mathbb{R}^{k_\beta}$ induced by the symbol pattern in row β of P. Because B is the column replacement of $A_1, A_2, \ldots, A_m$ into P, it follows from $(B_\beta)\mathbf{z} = \mathbf{0}$ that $A_\beta \mathbf{w} = \mathbf{0}$. We now show that $\mathbf{w} = \mathbf{0}$.

For $1 \leqslant \sigma \leqslant k_\beta$, let

$$s_\sigma^+ = \sum\{|z_\gamma|\colon\ \gamma \in C,\ z_\gamma > 0,\ p_{\beta\gamma} = \sigma\},$$
$$s_\sigma^- = \sum\{|z_\gamma|\colon\ \gamma \in C,\ z_\gamma < 0,\ p_{\beta\gamma} = \sigma\},$$
$$\bar{s}_\sigma^+ = \sum\{|z_\gamma|\colon\ \gamma \in \overline{C},\ z_\gamma > 0,\ p_{\beta\gamma} = \sigma\},$$
$$\bar{s}_\sigma^- = \sum\{|z_\gamma|\colon\ \gamma \in \overline{C},\ z_\gamma < 0,\ p_{\beta\gamma} = \sigma\}.$$

Let $D = \{p_{\beta\gamma}\colon\ \gamma \in C\}$ (note that $|D| \leqslant q_\beta$) and let $\overline{D} = \{1, \ldots, k_\beta\} \setminus D$. Because $\|\mathbf{z}_{|C}\|_1 - \|\mathbf{z}_{|\overline{C}}\|_1 \geqslant \mathbf{0}$,

$$\sum_{\sigma \in D} \left(s_\sigma^+ + s_\sigma^- - \bar{s}_\sigma^+ - \bar{s}_\sigma^-\right) - \sum_{\sigma \in \overline{D}} \left(\bar{s}_\sigma^+ + \bar{s}_\sigma^-\right) \geqslant 0. \tag{1}$$

Now $w_\sigma = s_\sigma^+ - s_\sigma^- + \bar{s}_\sigma^+ - \bar{s}_\sigma^-$ for $1 \leqslant \sigma \leqslant k_\beta$. If $\sigma \in D$ then our choice of β guarantees that either $s_\sigma^+ = 0$ or $s_\sigma^- = 0$ and hence $|w_\sigma| \geqslant s_\sigma^+ + s_\sigma^- - \bar{s}_\sigma^+ - \bar{s}_\sigma^-$. If $\sigma \in \overline{D}$ then $s_\sigma^+ = s_\sigma^- = 0$, so $|w_\sigma| \leqslant \bar{s}_\sigma^+ + \bar{s}_\sigma^-$. Then by (1),

$$\sum_{\sigma \in D} |w_\sigma| - \sum_{\sigma \in \overline{D}} |w_\sigma| \geqslant 0. \tag{2}$$

However, because $|D| \leqslant q_\beta$, $A_\beta \mathbf{w} = \mathbf{0}$, and A_β meets the (ℓ_1, q_β)-null space condition, either the left-hand side of (2) is negative or $\mathbf{w} = \mathbf{0}$. Thus $\mathbf{w} = \mathbf{0}$. We now show that $\mathbf{z} = \mathbf{0}$, which contradicts our assumptions.

Because $|w_\sigma| \geqslant s_\sigma^+ + s_\sigma^- - \bar{s}_\sigma^+ - \bar{s}_\sigma^-$ for each $\sigma \in D$, and $\mathbf{w} = \mathbf{0}$, we have that $s_\sigma^+ + s_\sigma^- - \bar{s}_\sigma^+ - \bar{s}_\sigma^- \leqslant 0$ for each $\sigma \in D$. So by (1), $\bar{s}_\sigma^+ + \bar{s}_\sigma^- = 0$ for each $\sigma \in \overline{D}$. Hence

$$\bar{s}_\sigma^+ = \bar{s}_\sigma^- = 0 \quad \text{for each } \sigma \in \overline{D}. \tag{3}$$

Similarly by (1), $s_\sigma^+ + s_\sigma^- - \bar{s}_\sigma^+ - \bar{s}_\sigma^- = 0$ for each $\sigma \in D$. Combining this with $s_\sigma^+ - s_\sigma^- + \bar{s}_\sigma^+ - \bar{s}_\sigma^- = w_\sigma = 0$ for $1 \leqslant \sigma \leqslant k_\beta$, it follows that

$$s_\sigma^+ = \bar{s}_\sigma^- \quad \text{and} \quad s_\sigma^- = \bar{s}_\sigma^+ \quad \text{for each } \sigma \in D. \tag{4}$$

Let $\tau = p_{\beta\chi}$. By our choice of β and because $z_\chi \geqslant 0$, we have that $s_\tau^- = 0$. Thus, by (3) when $\tau \in \overline{D}$ and by (4) when $\tau \in D$, $\bar{s}_\tau^+ = 0$. Because $z_\chi \geqslant 0$ and $\chi \in \overline{C}$, we have $0 = \bar{s}_\tau^+ \geqslant z_\chi \geqslant 0$ and hence that $z_\chi = 0$. So, from the definition of χ, $z_\gamma = 0$ for each $\gamma \in \overline{C}$, and thus it follows from (4) that $\mathbf{z} = \mathbf{0}$. □

3.4. Column replacement and RIP parameters

For vectors $\mathbf{a_1}, \ldots, \mathbf{a_s}$ (of any dimensions) the vector $\mathbf{a}$ formed by concatenating $\mathbf{a_1}, \ldots, \mathbf{a_s}$ satisfies $(\|\mathbf{a}\|_p)^p = (\|\mathbf{a_1}\|_p)^p + \cdots + (\|\mathbf{a_s}\|_p)^p$ whenever p is a nonnegative real.

The *binary expansion* $B(P)$ of an $\mathsf{HF}(m; k, \mathbf{v})$, P, is a $(\sum_{i=1}^m v_i) \times k$ array with entries from $\{0, 1\}$, obtained by taking the column replacement of $I_{v_1}, \ldots, I_{v_m}$ into P, where I_{v_i} is the $v_i \times v_i$ identity matrix. Recall the definition of the dth RIP parameter of a matrix from Section 3.1.

Lemma 3.7. *Let $A \in \mathbb{R}^{r \times k}$, $A = (a_{ij})$, be a measurement matrix, $P \in \{1, \ldots, k\}^{m \times n}$, $P = (p_{ij})$, be a pattern matrix, $B(P)$ be the binary expansion of P, and B be the column replacement of A into P. Then, for $1 \leqslant t \leqslant r$,*

$$\delta_t(B) \leqslant \frac{\delta_t(A) + \delta_t(B(P))}{1 + \delta_t(A)\delta_t(B(P))}.$$

Proof. Let $\mathbf{z} \in \mathbb{R}^n$ be a t-sparse vector. For $1 \leqslant \beta \leqslant m$, let $\mathbf{w}_\beta$ be the vector formed by setting $(w_\beta)_\sigma = \sum\{z_\gamma\colon\ p_{\beta\gamma} = \sigma,\ 1 \leqslant \gamma \leqslant n\}$ for $1 \leqslant \sigma \leqslant k$ ($\mathbf{w}_\beta$ can be considered as a projection of $\mathbf{z}$ onto $\mathbb{R}^k$ induced by the symbol pattern in row β of P). The vector $B\mathbf{z}$ is the concatenation of the vectors $A\mathbf{w}_1, \ldots, A\mathbf{w}_m$, and hence

$$\left(\|B\mathbf{z}\|_2\right)^2 = \left(\|A\mathbf{w}_1\|_2\right)^2 + \cdots + \left(\|A\mathbf{w}_m\|_2\right)^2. \tag{5}$$

Using the RIP properties of A, for some $R > 0$ and for $1 \leqslant \beta \leqslant m$,

$$\left(1 - \delta_t(A)\right)R\left(\|\mathbf{w}_\beta\|_2\right)^2 \leqslant \left(\|A\mathbf{w}_\beta\|_2\right)^2 \leqslant \left(1 + \delta_t(A)\right)R\left(\|\mathbf{w}_\beta\|_2\right)^2.$$

Substituting into (5),

$$\left(1 - \delta_t(A)\right)R\left(\left(\|\mathbf{w}_1\|_2\right)^2 + \cdots + \left(\|\mathbf{w}_m\|_2\right)^2\right) \leqslant \left(\|B\mathbf{z}\|_2\right)^2 \leqslant \left(1 + \delta_t(A)\right)R\left(\left(\|\mathbf{w}_1\|_2\right)^2 + \cdots + \left(\|\mathbf{w}_m\|_2\right)^2\right).$$

Now $B(P)\mathbf{z}$ is the concatenation of the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$ and hence

$$\left(\left\|B(P)\mathbf{z}\right\|_2\right)^2 = \left(\|\mathbf{w}_1\|_2\right)^2 + \cdots + \left(\|\mathbf{w}_m\|_2\right)^2.$$

Using the RIP properties of $B(P)$, for some $R' > 0$

$$\left(1 - \delta_t\left(B(P)\right)\right)R'\left(\|\mathbf{z}\|_2\right)^2 \leqslant \left(\|\mathbf{w}_1\|_2\right)^2 + \cdots + \left(\|\mathbf{w}_m\|_2\right)^2 \leqslant \left(1 + \delta_t\left(B(P)\right)\right)R'\left(\|\mathbf{z}\|_2\right)^2$$

and hence

$$\left(1 - \delta_t(A)\right)\left(1 - \delta_t\left(B(P)\right)\right)RR'\left(\|\mathbf{z}\|_2\right)^2 \leqslant \left(\|B\mathbf{z}\|_2\right)^2 \leqslant \left(1 + \delta_t(A)\right)\left(1 + \delta_t\left(B(P)\right)\right)RR'\left(\|\mathbf{z}\|_2\right)^2.$$

Finally, let $R'' = (1 + \delta_t(A)\delta_t(B(P)))RR'$. Substituting for RR' in both the upper and lower bounds and simplifying,

$$\left(1 - \frac{\delta_t(A) + \delta_t(B(P))}{1 + \delta_t(A)\delta_t(B(P))}\right)R''\left(\|\mathbf{z}\|_2\right)^2 \leqslant \left(\|B\mathbf{z}\|_2\right)^2 \leqslant \left(1 + \frac{\delta_t(A) + \delta_t(B(P))}{1 + \delta_t(A)\delta_t(B(P))}\right)R''\left(\|\mathbf{z}\|_2\right)^2$$

and the result follows. □

RIP has been generalized to norms other than the ℓ_2 norm [4]. Let p be a positive integer. For a matrix A with real entries, the *t-th RIP-p parameter* of A, denoted $\delta_{p,t}(A)$, is the smallest real number δ such that, for some real number $R > 0$, $(1 - \delta)R(\|\mathbf{x}\|_p)^p \leqslant (\|A\mathbf{x}\|_p)^p \leqslant (1 + \delta)R(\|\mathbf{x}\|_p)^p$ for all t-sparse vectors $\mathbf{x}$. The case when $p = 2$ is the standard notion of RIP. RIP-1 parameters are used in studying the adjacency matrices of expander graphs [4,41]. Matrix A is *normalized* for its t-th RIP-p parameter if $(1 - \delta_{p,t}(A))(\|\mathbf{x}\|_p)^p \leqslant (\|A\mathbf{x}\|_p)^p \leqslant (1 + \delta_{p,t}(A))(\|\mathbf{x}\|_p)^p$ for all t-sparse vectors $\mathbf{x}$. Any matrix can be normalized in this way by multiplying it by a scalar.

Lemma 3.8. *For $1 \leqslant i \leqslant m$, let $A_i \in \mathbb{R}^{r_i \times k_i}$ be a measurement matrix that is normalized for its t-th RIP-p parameter, and let $P = (p_{ij})$ be an $m \times n$ pattern matrix in which row β contains symbols from $\{1, \ldots, k_\beta\}$ for $1 \leqslant \beta \leqslant m$. Let $B(P)$ be the binary expansion of P, and B be the column replacement of $A_1, \ldots, A_m$ into P. Then, for any positive integer p and for $1 \leqslant t \leqslant r$,*

$$\delta_{p,t}(B) \leqslant \frac{\delta_{\max} + \delta_{p,t}(B(P))}{1 + \delta_{\max}\delta_{p,t}(B(P))},$$

where $\delta_{\max} = \max(\delta_{p,t}(A_1), \ldots, \delta_{p,t}(A_m))$.

The proof parallels that of Lemma 3.7 closely, and is omitted. These two lemmas provide some evidence that column replacement enables us not just to meet recoverability conditions, but also to preserve the basic machinery to deal with noise in the signal.

4. Strengthening hash families using the Lovász local lemma

The Lovász local lemma provides a powerful tool for establishing existence of combinatorial arrays; see [28] for applications to certain hash families. We use the following version (as usual, e is the base of the natural logarithm).

Theorem 4.1. *(See [33].) Suppose that $A_1, \ldots, A_\ell$ are events in a probability space such that each event is mutually independent of a set of at least $\ell - b$ other events. If $\Pr[A_i] \leqslant \frac{1}{e(b+1)}$ for $1 \leqslant i \leqslant \ell$, then*

$$\Pr\left[\bigcap_{i=1}^{\ell} \overline{A_i}\right] > 0.$$

To simplify the analysis we only consider homogeneous hash families in this section, and use certain notation throughout.

Let $\mathcal{W}_{t,s}$ be the set of all s-element multisets whose entries are nonnegative integers that sum to t. For $W = \{w_1, \ldots, w_s\} \in \mathcal{W}_{t,s}$, let $U(W)$ be the total number of unordered partitions of a set of size t into sets of sizes $w_1, \ldots, w_s$. Then

$$U(W) = \frac{1}{\prod_{i \in \mathbb{Z}} \mu(i)!}\binom{t}{w_1}\binom{t - w_1}{w_2}\binom{t - w_1 - w_2}{w_3}\cdots\binom{w_s}{w_s},$$

where $\mu(i)$ is the multiplicity of i in the multiset W. Let $S(t, s)$ be the number of unordered partitions of a set of size t into s non-empty subsets. This is a Stirling number of the second kind and can be calculated as

$$S(t, s) = \frac{1}{s!}\sum_{i=0}^{s}(-1)^i\binom{s}{i}(s - i)^t.$$

Let $\mathcal{P}_v = \{\mathbf{p} = (p_1, \ldots, p_v)\colon\ p_i \in \mathbb{R}_{\geqslant 0}$ for $1 \leqslant i \leqslant v$ and $\sum_{i=1}^{v} p_i = 1\}$.

Let d be a positive integer and let $\mathbf{p} = (p_1, \ldots, p_v) \in \mathcal{P}_v$. Suppose the vertices of a complete multipartite graph with parts of sizes $W = \{w_1, w_2, \ldots, w_s\}$ are coloured independently at random with colours from the set $\{1, \ldots, v\}$ so that the probability of any specified vertex receiving colour i is p_i for $1 \leqslant i \leqslant v$. The probability that this process results in a proper d-colouring of the graph is denoted by $\pi(W, \mathbf{p}, d, v)$. For $W \in \mathcal{W}_{t,s}$, let $\pi_{\mathsf{S}}(W, d, v) = \max_{\mathbf{p}\in\mathcal{P}_v} \pi(W, \mathbf{p}, d, v)$. Let $\pi_{\mathsf{D}}(t, s, d, v) = \max_{\mathbf{p}\in\mathcal{P}_v} \min_{W\in\mathcal{W}_{t,s}} \pi(W, \mathbf{p}, d, v)$.

Theorem 4.2. *Let $w_1, \ldots, w_s$ be nonnegative integers, and $t = \sum_{i=1}^{s} w_i$. There exists*

1. *a d-strengthening $\mathsf{SHF}(N; k, v, \{w_1, \ldots, w_s\})$ if*

$$N \geqslant \frac{\log(e \cdot U(W)(\binom{k}{t} - \binom{k-t}{t}))}{-\log(1 - \pi_{\mathsf{S}}(W, d, v))};$$

2. *a d-strengthening $\mathsf{DHF}(N; k, v, t, s)$ if*

$$N \geqslant \frac{\log(e \cdot S(t, s)(\binom{k}{t} - \binom{k-t}{t}))}{-\log(1 - \pi_{\mathsf{D}}(t, s, d, v))}.$$

Proof. Let K be a set of k column indices. Let $\mathcal{A} = \{\{Z_1, \ldots, Z_s\}\colon\ Z_i \subset K \text{ for } 1 \leqslant i \leqslant s,\ \ Z_i \cap Z_j = \emptyset \text{ for } 1 \leqslant i < j \leqslant s, |\bigcup_{i=1}^{s} Z_i| = t\}$. For $W = \{w_1, \ldots, w_s\}$ let $\mathcal{A}_W = \{\{Z_1, \ldots, Z_s\} \in \mathcal{A}\colon\ |Z_i| = w_i \text{ for } 1 \leqslant i \leqslant s\}$. In the remainder, the variables have the following interpretations:

Variable	First statement	Second statement
$\mathcal{Z}$	$\mathcal{A}_W$	$\mathcal{A}$
C	$U(W)$	$S(t, s)$
f	$\pi_{\mathsf{S}}(W, d, v)$	$\pi_{\mathsf{D}}(t, s, d, v)$
$\mathbf{p} = (p_1, \ldots, p_v)$	$\mathbf{p} \in \mathcal{P}_v$ satisfies $\pi(W, \mathbf{p}, d, v) = f$	$\mathbf{p} \in \mathcal{P}_v$ satisfies $\min_{W\in\mathcal{W}_{t,s}} \pi(W, \mathbf{p}, d, v) = f$

Let H be an $N \times k$ matrix whose entries are chosen independently at random from the set $\{1, \ldots, v\}$ so that the probability of any specified entry being i is p_i for $i = 1, \ldots, v$. For $\{Z_1, \ldots, Z_s\} \in \mathcal{Z}$, let $A_{\{Z_1,\ldots,Z_s\}}$ be the event that there is no row of H in which at most d distinct symbols occur in the columns in $Z_1 \cup \cdots \cup Z_s$ and in which the sets of symbols in the columns in $Z_1, \ldots, Z_s$ are pairwise disjoint. Array H is the required hash family if and only if none of the events in $\{A_{\{Z_1,\ldots,Z_s\}}\colon\ \{Z_1, \ldots, Z_s\} \in \mathcal{Z}\}$ occurs. It can be seen that $\Pr[A_{\{Z_1,\ldots,Z_s\}}] \leqslant (1 - f)^N$.

An event $A_{\{Z_1,\ldots,Z_s\}}$ is mutually independent of all events in $\{A_{\{Z'_1,\ldots,Z'_s\}}\colon\ \{Z'_1, \ldots, Z'_s\} \in \mathcal{Z}$ and $(Z_1 \cup \cdots \cup Z_s) \cap (Z'_1 \cup \cdots \cup Z'_s) = \emptyset\}$; there are $\binom{k-t}{t}C$ such events. Because there are $\binom{k}{t}C$ events in $\{A_{\{Z_1,\ldots,Z_s\}}\colon\ \{Z_1, \ldots, Z_s\} \in \mathcal{Z}\}$, by Theorem 4.1 the required hash family exists if $(1 - f)^N \leqslant \frac{1}{e(b+1)}$, where $b = \binom{k}{t}C - \binom{k-t}{t}C - 1$. Rearrange to produce the required results. □

In order to develop deterministic, efficient algorithms, we use a consequence of [14, Theorem 1.2], corresponding to the special case of that result in which each event is mutually independent of a set of all but at most b other events. In the notation of [14], ϵ is arbitrarily close to 1 and the probability of each event is assigned the value $\frac{1}{b+1}$.

Theorem 4.3. *(See [14].) Let $\mathcal{X} = \{X_1, \ldots, X_\gamma\}$ be a collection of mutually independent discrete random variables each with a domain of cardinality at most v. Let $\mathcal{A} = \{A_1, \ldots, A_\ell\}$ be a collection of events such that each event is determined by at most r of the variables in $\mathcal{X}$ and each event is mutually independent of a set of at least $\ell - b$ other events. Suppose that time T is sufficient to compute the conditional probability $\Pr[A|X_i = \sigma_i$ for each $i \in I]$ for any event $A \in \mathcal{A}$, any $I \subseteq \{1, \ldots, \gamma\}$, and any partial evaluation $\{\sigma_i\}_{i\in I}$ of the variables in $\mathcal{X}$. If $\Pr[A] \leqslant (\frac{1}{e(b+1)})^2$ for all $A \in \mathcal{A}$, then an evaluation of $X_1, \ldots, X_\gamma$ under which none of the events in $\mathcal{A}$ occurs can be found by a deterministic algorithm running in time $O(T \cdot r \cdot M^{4+\delta})$, where δ is any positive real number, and $M = \max(\gamma, \ell, \frac{\ell r}{b})$.*

Theorem 4.4. *Let $v, w_1, \ldots, w_s$, and d be constant nonnegative integers. Let $t = \sum_{i=1}^{s} w_i$. There are deterministic algorithms running in time $O(k^{4t+\delta})$ for any positive real number δ to construct*

1. *a d-strengthening $\mathsf{SHF}(N; k, v, \{w_1, \ldots, w_s\})$ where*

$$N = \left\lceil \frac{2\log(e \cdot U(W)(\binom{k}{t} - \binom{k-t}{t}))}{-\log(1 - \pi_{\mathsf{S}}(W, d, v))} \right\rceil;$$

2. *a d-strengthening* $\mathsf{DHF}(N; k, v, t, s)$ *where*

$$N = \left\lceil \frac{2\log(e \cdot S(t,s)(\binom{k}{t} - \binom{k-t}{t}))}{-\log(1 - \pi_{\mathsf{D}}(t,s,d,v))} \right\rceil.$$

Proof. Let $\mathbf{p}$, H, f, C, $\mathcal{Z}$ and the events $\{A_{\{Z_1,\ldots,Z_s\}}\colon \{Z_1,\ldots,Z_s\} \in \mathcal{Z}\}$ be as defined in the proof of Theorem 4.2. Again, H is the required hash family if and only if none of the events $\{A_{\{Z_1,\ldots,Z_s\}}\colon \{Z_1,\ldots,Z_s\} \in \mathcal{Z}\}$ occur. Once more, $\Pr[A_{\{Z_1,\ldots,Z_s\}}] \leqslant (1-f)^N$ for each event $A_{\{Z_1,\ldots,Z_s\}}$; there are $\binom{k}{t}C$ events in total; and each event is mutually independent of a set of at least $\binom{k-t}{t}C$ other events.

Applying Theorem 4.3, considering the entries of H to be discrete random variables, we can obtain a deterministic algorithm for constructing the required hash family provided that $(1-f)^N \leqslant (\frac{1}{e(b+1)})^2$, where $b = \binom{k}{t}C - \binom{k-t}{t}C - 1$. This holds for the given value of N.

We now bound the run time. There are Nk entries in H, and each event depends on Nt entries of H. For sufficiently large k we have $Nk = O(k\log k)$, $\binom{k}{t}C = O(k^t)$ and, because $\frac{1}{b}\binom{k}{t}C = O(k)$, we have $\frac{Nt}{b}\binom{k}{t}C = O(k\log k)$. Thus $M = O(k^t)$ and $r = O(N) = O(\log k)$, where M and r are as in Theorem 4.3.

Let $\{Z_1,\ldots,Z_s\} \in \mathcal{Z}$. Let H' be a partial evaluation of the entries of H. For a specific row β of H, it takes time $O(1)$ to compute the conditional probability, given the partial evaluation H', that at most d distinct symbols occur in the columns in $Z_1 \cup \cdots \cup Z_s$ of row β and the sets of symbols in the columns in $Z_1,\ldots,Z_s$ of row β are pairwise disjoint (the computation involves considering at most t entries each of which can take at most v values, and both are constant). Thus the conditional probability of $A_{\{Z_1,\ldots,Z_s\}}$ given H' can be computed in time $O(N)$. Thus $T = O(N) = O(\log k)$, where T is as in Theorem 4.3. Combining these bounds yields the bound we require. □

5. The Stein–Lovász–Johnson method

Next we examine a different probabilistic approach that yields a class of straightforward greedy construction methods for which the size of the hash family produced meets the logarithmic bound. Indeed, after some substantial preliminaries, we establish that it provides an efficient algorithm for hash families of fixed strength. To develop this, we detour into set cover problems.

5.1. Set cover problems

Let X be a finite set of size n, and let $\mathcal{D}$ be a collection of subsets of X. A *set cover* for the set system $(X, \mathcal{D})$ is a collection $\mathcal{D}' \subseteq \mathcal{D}$ so that $\bigcup_{B\in\mathcal{D}'} B = X$. Finding the smallest set cover for a set system is NP-hard [43], and hence approximation and heuristic techniques have been developed. Stein [51], Lovász [44], and Johnson [42] (see also [16]) analyze a greedy algorithm and establish a useful upper bound on the sizes of set covers that it produces. At a high level, the algorithm repeatedly selects one set for inclusion in the set cover which, once selected, is never removed. The selection of the set is greedy, choosing one that covers the largest number of uncovered elements at that stage, and breaking ties arbitrarily. We give (one version of) the algorithm in Algorithm 1.

Algorithm 1 The Greedy Algorithm

```
GREEDY_SET_COVER(X, D): (|X| = n; |D| = c)
    Set r(x) = |{B: x ∈ B ∈ D}| for x ∈ X
    Set α = max{|B|: B ∈ D} and r = min{r(x) : x ∈ X}
    Set M_j = ∅ for 0 ⩽ j ⩽ α
    Set D_0 = D and Y_0 = X
    Set n_α = |X|
    Set L = ∅ and i = 0
    while Y_i ≠ ∅ do
        Set γ_i = |D_i|; ρ_i = |Y_i|; and α_i = max{|B|: B ∈ D_i}
        If i > 0 and α_i < α_{i-1} set n_{α_i} = |Y_i|
        Choose a set D_i ∈ D_i for which |D_i| = α_i
        Set L = L ∪ {D_i}
        Set M_{α_i} = M_{α_i} ∪ {D_i}
        Set Y_{i+1} = Y_i \ D_i
        Set D_{i+1} = {B \ D_i: B ∈ D_i, B ⊈ D_i}
        Set i = i + 1
    Set n_0 = 0
    return L
```

The representation of a hash family as a set cover is straightforward: The sets are in one-to-one correspondence with the possible rows, the elements consist of all required separations of columns, and a set contains an element precisely when

the corresponding row carries out the specified separation. A set cover is then a set of rows which together accomplish all of the required separations. Hence Greedy_Set_Cover provides an algorithm for producing hash families.

We use the notation introduced in Algorithm 1: n is the number of elements, c is the number of sets, α is the size of the largest set, and r is the smallest number of sets to which an element belongs. The size of the set cover can be viewed either as the smallest value of i for which no elements remain to be covered ($Y_i = \emptyset$), or it can be viewed as $\sum_{j=0}^{\alpha} \ell_j$, taking $\ell_j = |\mathcal{M}_j|$. We consider the latter expression. First, $n_{j-1} = n_j - j\ell_j$, and hence $\ell_j = (n_j - n_{j-1})/j$. Because in each set system considered when choosing the sets in $\mathcal{M}_j$, every element appears in at least r sets and every set has size at most j, $rn_j \leqslant jc$. Moreover,

$$\ell = \sum_{j=1}^{\alpha} \ell_j = \sum_{j=1}^{\alpha} \frac{n_j - n_{j-1}}{j} = \frac{n_\alpha}{\alpha} + \frac{n_{\alpha-1}}{\alpha(\alpha-1)} + \frac{n_{\alpha-2}}{(\alpha-1)(\alpha-2)} + \cdots + \frac{n_1}{2 \cdot 1} - n_0.$$

Combining these, we obtain $\ell \leqslant \frac{n}{\alpha} + \frac{c}{r}(\sum_{j=2}^{\alpha} \frac{1}{j})$, which yields the bound in the theorem of Stein [51], Lovász [44], and Johnson [42]:

Theorem 5.1 *(Stein–Lovász–Johnson). Let $(X, \mathcal{D})$ be a set system with $|X| = n$ and $|\mathcal{D}| = c$ so that $|B| \leqslant \alpha$ for every $B \in \mathcal{D}$ and $|\{B\colon x \in B \in \mathcal{D}\}| \geqslant r$ for every $x \in X$. Then there is a collection $\mathcal{D}' \subseteq \mathcal{D}$ forming a set cover with ℓ sets for some $\ell \leqslant \frac{n}{\alpha} + \frac{c}{r} \ln \alpha \leqslant \frac{c}{r}(1 + \ln \alpha)$.*

A second analysis of Greedy_Set_Cover uses the fact that it terminates when $Y_i = \emptyset$, or equivalently when $\rho_i < 1$. An element x of Y_i is not a member of $\bigcup_{j=0}^{i-1} D_j$, and hence $r(x)$ is unchanged by the deletion of the elements already covered. Because $r(x) \geqslant r$ for all $x \in Y_i$, and $\gamma_i \leqslant c$, the size of D_i is at least $\frac{r\rho_i}{c}$. Then $\rho_{i+1} \leqslant \rho_i - \frac{r\rho_i}{c} = \rho_i \frac{c-r}{c}$. Because this holds for every $i > 0$, $\rho_i \leqslant n(\frac{c-r}{c})^i$. We determine the smallest value of i for which $n(\frac{c-r}{c})^i < 1$. Equivalently, $n < (\frac{c}{c-r})^i$. Taking logarithms base $c/(c-r)$ of both sides, $\log_{c/(c-r)} n < i$.

This establishes:

Theorem 5.2. *Let $(X, \mathcal{D})$ be a set system with $|X| = n$ and $|\mathcal{D}| = c$ so that $|\{B\colon x \in B \in \mathcal{D}\}| \geqslant r$ for every $x \in X$. Then there is a collection $\mathcal{D}' \subseteq \mathcal{D}$ forming a set cover with ℓ sets for some $\ell \leqslant 1 + \frac{\log n}{\log(c/(c-r))}$.*

This improves the constant in the bound over that of the Stein–Lovász–Johnson theorem in some cases, but yields a weaker bound in others. This apparent discrepancy is an artifact of the analyses, not the algorithm.

The Stein–Lovász–Johnson method has been applied in establishing upper bounds on the sizes of numerous combinatorial arrays. In these contexts, however, the admissible sets $\mathcal{D}$ are often known implicitly rather than presented explicitly. Then the size of the input is simply $|X|$, and the run time of the greedy algorithm may be exponential in $|X|$, because $|\mathcal{D}|$ may be exponentially large with respect to $|X|$. This has limited the practical uses of such greedy methods for the actual construction of the set covers needed to produce the corresponding combinatorial arrays.

Careful examination shows that the only operations in Algorithm 1 that consider all of the sets in $\mathcal{D}_i$ are the ones to select D_i, and to remove all elements of D_i from the sets in $\mathcal{D}_i$ to form $\mathcal{D}_{i+1}$. To obtain an algorithm whose running time is polynomial in $|X|$, we cannot hope to examine (or even list) all sets in $\mathcal{D}$. Our first task, then, is to simplify the selection of the set D_i. In fact, we show that selecting a set of *average* size yields the same results. At the same time, we equip $\mathcal{D}$ with a probability distribution, so that $\Pr[B]$ is the probability that set $B \in \mathcal{D}$ is selected.

Average_Set_Cover, shown in Algorithm 2, is essentially the same method – with one important difference. Each set selected is only required to cover the average number of as yet uncovered elements of X rather than the maximum. Moreover, this average is weighted by the initial probability distribution selected on $\mathcal{D}$.

The analyses of Greedy_Set_Cover carry through for the average method as well. For the first, we employed the fact that $n_{j-1} = n_j - j\ell_j$, and hence $\ell_j = (n_j - n_{j-1})/j$; and the fact that $rn_j \leqslant jc$. For the average method, $n_{j-1} \leqslant n_j - j\ell_j$, and hence $\ell_j \leqslant (n_j - n_{j-1})/j$; and $rn_j \leqslant jc$ because $j = \lceil \frac{rn_j}{c} \rceil$. For the second, we employed only the fact that $|D_i| \geqslant \frac{r\rho_i}{c}$, which holds for the average method as well.

Hence

Theorem 5.3. *Let $(X, \mathcal{D})$ be a set system with $|X| = n$ and $|\mathcal{D}| = c$, for which $\Pr[B]$ is the probability that $B \in \mathcal{D}$ is selected. Let $r(x) = c \sum_{\{B \in \mathcal{D}\colon x \in B\}} \Pr[B]$ and $r = \min\{r(x)\colon x \in X\}$. Let $\beta = \lceil \frac{\sum_{x \in X} r(x)}{c} \rceil$. Then Average_Set_Cover produces a set cover $\mathcal{D}' \subseteq \mathcal{D}$ with ℓ sets, where $\ell \leqslant \min\left(\frac{c}{r}(1 + \ln \beta), 1 + \frac{\log n}{\log(c/(c-r))}\right)$.*

When the probability distribution is uniform (and in many other cases), $\beta \leqslant \alpha$, and hence Theorem 5.3 improves on Theorem 5.1. This may come as a surprise, because the original Stein–Lovász–Johnson method selects a largest set while the average method may select a smaller one. Again, the discrepancy arises from the algorithm analyses, not from the algorithms themselves. In practice, it is quite possible that selecting the maximum coverage yields a better result in the

Algorithm 2 The Average Algorithm

AVERAGE_SET_COVER($X, \mathcal{D}$): ($|X| = n$; $|\mathcal{D}| = c$)
  Set $r(x) = c \sum_{\{B \in \mathcal{D}:\ x \in B\}} \Pr[B]$ for $x \in X$
  Set $r = \min\{r(x):\ x \in X\}$ and $\alpha = \lceil \frac{\sum_{x \in X} r(x)}{c} \rceil$
  Set $\mathcal{M}_j = \emptyset$ for $0 \leqslant j \leqslant \alpha$
  Set $\mathcal{D}_0 = \mathcal{D}$ and $Y_0 = X$
  Set $n_\alpha = |X|$
  Set $\mathcal{L} = \emptyset$ and $i = 0$
  while $Y_i \neq \emptyset$ **do**
    Set $\gamma_i = |\mathcal{D}_i|$; $\rho_i = |Y_i|$; and $\alpha_i = \lceil \frac{\sum_{x \in Y_i} r(x)}{c} \rceil$
    If $i > 0$ and $\alpha_i < \alpha_{i-1}$ set $n_{\alpha_i} = |Y_i|$
    Choose a set $D_i \in \mathcal{D}_i$ for which $|D_i| \geqslant \alpha_i$ and $\Pr[D_i] > 0$
    Set $\mathcal{L} = \mathcal{L} \cup \{D_i\}$
    Set $\mathcal{M}_{\alpha_i} = \mathcal{M}_{\alpha_i} \cup \{D_i\}$
    Set $Y_{i+1} = Y_i \setminus D_i$
    Set $\mathcal{D}_{i+1} = \{B \setminus D_i:\ B \in \mathcal{D}_i,\ B \not\subseteq D_i\}$
    Set $i = i + 1$
  Set $n_0 = 0$
  return $\mathcal{L}$

end than does selecting the average coverage. Nevertheless, Theorem 5.3 shows that the *conclusion* of Theorem 5.1 can be obtained by selecting sets with average coverage.

5.2. The application to hash families

Despite the exponential run time exhibited by AVERAGE_SET_COVER when applied directly to hash families, the Stein–Lovász–Johnson paradigm has been used to develop methods whose run time is polynomial in $|X|$. One method is to list only a small subset of the sets [18]. Another method employs an implicit representation of all of the sets; see, for example, [7,8] for an application to the construction of 'covering arrays.' We first outline the idea using perfect hash families, adapting a method from [19].

We construct a $\mathsf{PHF}(N; k, v, t)$ with entries chosen from an alphabet Σ. Take K to be a set of k column indices and $X = \binom{K}{t}$. Form a set $\mathcal{D}$ of v^k subsets of X, one for each k-tuple in Σ^k. The set corresponding to the k-tuple $(x_1, \ldots, x_k) \in \Sigma^k$ contains element $\{\gamma_1, \ldots, \gamma_t\}$ exactly when $|\{x_{\gamma_i}:\ 1 \leqslant i \leqslant t\}| = t$. Then every element of X belongs to $v^{k-t} \cdot v \cdot (v-1) \cdot \cdots \cdot (v-t+1)$ sets in $\mathcal{D}$. Every set in $\mathcal{D}$ covers at most $\binom{k}{t}$ elements of X, and hence by the Stein–Lovász–Johnson theorem we find that $N \leqslant 1 + \frac{v^t}{v(v-1)\cdots(v-t+1)} \ln \binom{k}{t}$. This yields an exponential time method. By showing that it suffices to find a set that covers an *average* number of uncovered elements and developing a method of conditional expectations to find such a set, Colbourn [19] developed an efficient (time polynomial in k for fixed t) algorithm for the construction of perfect hash families with a number of rows meeting the given bound.

Whether the sets are listed explicitly or not, one potential benefit of selecting a set with average rather than maximum coverage is that the average can often be easily computed or bounded, and then finding any set with at least that average coverage suffices. We explore this next.

There is evidently a wide variety of possible conditions that might be imposed on the hash family to be constructed. To treat these variants, we proceed as follows. Consider creating a $\mathsf{HF}(N; k, \mathbf{v})$, where the symbols of row ρ are chosen from an alphabet Σ_ρ of size v_ρ. We define a number of *requirements* $R_1, \ldots, R_q$, each of which is a partition $C_1, \ldots, C_s$ of a set C of at most t columns of the hash family. An *assignment* A to the requirement R for row ρ is an assignment of symbols from Σ_ρ to the columns associated with R. For $1 \leqslant \rho \leqslant N$, we define a *constraint function* P_ρ which accepts a requirement R and an assignment A to R and outputs a value in $\{\mathbf{true}, \mathbf{false}\}$. Any k-tuple $\mathbf{x} \in \Sigma_\rho^k$, representing a possible row ρ, induces an assignment to a requirement R. We represent this assignment, along with the requirement, by $A_{\mathbf{x},R}$. The hash family *satisfies* a requirement R if $P_\rho(A_{\mathbf{x},R}) = \mathbf{true}$ for some $1 \leqslant \rho \leqslant N$, where $\mathbf{x}$ is the k-tuple representing row ρ of the hash family. Thus, while the requirements for the hash family are fixed at the outset, the constraints for meeting each requirement may vary from row to row.

From Theorem 5.3, one can immediately deduce bounds on the sizes of hash families in a general setting. We suppose that the candidate rows are selected uniformly at random, that the hash family needed is homogeneous with v symbols, and that the constraint function is the same for each row. Take $c = v^k$ and $r = \mu v^k$ in Theorem 5.3 to establish:

Theorem 5.4. *An* $\mathsf{HF}(N; k, v)$ *satisfying* q *requirements exists whenever*

$$N \geqslant \min\left(\frac{1}{\mu}(1 + \ln \beta), 1 + \frac{\log q}{\log 1/(1-\mu)}\right),$$

taking δ_R *to be the ratio of the number of assignments* A *to* R *that satisfy the constraint for* R *to the total number of assignments to* R, μ *to be the minimum of* δ_R *over all requirements* R, *and* β *to be the ceiling of the sum of* δ_R *over all requirements* R.

In Theorem 5.4, it may be puzzling that the bound does not appear to involve k. However, when requirements are placed on all of the k columns, the number of requirements q must be a function of k, and β is a function of q. The quantities in Theorem 5.4 can often easily be calculated; we give one example. Suppose that $\mathcal{W} = \{\{1,4\},\{2,3\}\}$, and our objective is to produce a $\mathcal{W}$-separating, 3-scattering, and 3-strengthening $\mathsf{HF}(N;k,6)$. Write $K = k(k-1)(k-2)(k-3)(k-4)$. There are $\binom{k}{1}\binom{k-1}{4} = \frac{1}{24}K$ requirements for the $\{1,4\}$ separation, and $\binom{k}{2}\binom{k-2}{3} = \frac{1}{12}K$ for the $\{2,3\}$ separation. Each has $6^5 = 7776$ assignments. A $\{1,4\}$ separation R has 840 assignments that meet the constraint, so $\delta_R = \frac{35}{324}$. A $\{2,3\}$ separation R has 510 assignments that meet the constraint, so $\delta_R = \frac{85}{1296}$, $\mu = \frac{85}{1296}$, and $\beta = \frac{155}{15\,552}K$. Then the hash family exists provided that $N \geqslant \min\left(\frac{1296}{85}(1 + \ln\frac{155K}{15552}), 1 + \frac{\log\frac{K}{8}}{\log\frac{1296}{1211}}\right)$.

Following the paradigm of Average_Set_Cover, we proceed as follows. The set X is the set of all requirements, and N is (an upper bound on) the number of rows permitted. Then Average_Hash_Family$(X, N, \{P_\rho(A,R)\})$, given in Algorithm 3, produces the desired hash family, or may fail if N is too small. Provided that N satisfies the bound in Theorem 5.4, however, the algorithm is guaranteed to succeed.

Algorithm 3 The Average Algorithm for Hash Families

```
Average_Hash_Family(X, N, {P_ρ(A, R)})
    // {P_ρ(A, R)} provides a predicate for each row ρ, each R ∈ X,
    // and each assignment A to R
    Set X_1 = X and L = ∅
    for ρ from 1 to N do
        y = Select_Average_Row(ρ, X_ρ, {P_ρ(A, R)})
        Set L = L ∪ {y}
        Set X_{ρ+1} = X_ρ \ {R ∈ X_ρ: P_ρ(A_{y,R}) = true}
    if X_{N+1} = ∅ return L else return fail
```

Average_Hash_Family requires that we repeatedly select a next row for inclusion. For a requirement $R \in X_\rho$ and a candidate row $\mathbf{x} = (x_1, \ldots, x_k) \in \Sigma_\rho^k$, R is covered by the row exactly when $P_\rho(A_{\mathbf{x},R}) = \mathbf{true}$. For this reason, Select_Average_Row must find a row $\mathbf{x} \in \Sigma_\rho^k$ for which $\Pr[\mathbf{x}] > 0$ and $|\{R \in X_\rho\colon\ P_\rho(A_{\mathbf{x},R})\}|$ is at least the average over all choices of row $\mathbf{y}$.

Suppose that we simply selected the row $\mathbf{x}$ at random (according to the probability distribution) from Σ_ρ^k. Then $\Pr[\mathbf{x}] > 0$, and the expectation of $|\{R \in X_\rho\colon\ P_\rho(A_{\mathbf{x},R})\}|$ is precisely the desired average,

$$\sum_{\mathbf{x}\in\Sigma_\rho^k} \Pr[\mathbf{x}] \cdot \left(\sum_{R\in X_\rho} P_\rho(A_{\mathbf{x},R})\right) = \sum_{R\in X_\rho}\left(\sum_{\mathbf{x}\in\Sigma_\rho^k} \Pr[\mathbf{x}] \cdot P_\rho(A_{\mathbf{x},R})\right),$$

treating $P_\rho(A_{\mathbf{x},R})$ as a 0, 1-indicator variable. This yields a randomized algorithm for producing hash families.

6. Making the Stein–Lovász–Johnson method efficient

Here we establish that Average_Hash_Family (from Algorithm 3) provides a deterministic, polynomial-time algorithm when the maximum strength of the requirements is fixed. Consider the operation of that algorithm. It suffices to calculate the expectation of $P_\rho(A_{\mathbf{x},R})$ for each $R \in X_\rho$ in order to determine the average sought. Nevertheless, we must also find a row $\mathbf{x} \in \Sigma_\rho^k$ that yields at least this average. To do this, we start with a row in which no entries have been chosen, and repeatedly choose one coordinate whose entry is unspecified in which to choose an entry. Our objective is to ensure that at each stage the expectation of finding a row that covers at least the average number of requirements does not decrease. In other words, we want the conditional expectation, based on the selection of the entries already made, never to decrease. Hence we employ the fundamental idea in the *method of conditional expectations* [34,46].

Consider constructing an $\mathsf{HF}(N;k,\mathbf{v})$ in which the symbols in row ρ are selected from an alphabet Σ_ρ of size v_ρ. We must deal with rows in which only some of the entries have been chosen. Suppose that $(x_1,\ldots,x_k) \in (\Sigma_\rho \cup \{\star\})^k$. We interpret an entry in Σ_ρ to mean that the entry has been chosen, while the entry $\star$ means that the entry has not yet been chosen. A row $(y_1,\ldots,y_k) \in \Sigma_\rho^k$ is a *completion* of $(x_1,\ldots,x_k)$ if $x_i = y_i$ or $x_i = \star$ for $1 \leqslant i \leqslant k$. A row with s $\star$ entries has v_ρ^s completions, and the set of these is denoted by $\mathcal{R}_{\mathbf{x}}$. For two rows $\mathbf{x}$ and $\mathbf{y}$, the probability that $\mathbf{y}$ occurs given that $\mathbf{x}$ occurs, $\Pr[\mathbf{y}|\mathbf{x}]$, is 0 if $\sum_{\mathbf{z}\in\mathcal{R}_{\mathbf{x}}\cap\mathcal{R}_{\mathbf{y}}} \Pr[\mathbf{z}] = 0$; otherwise it is $(\sum_{\mathbf{z}\in\mathcal{R}_{\mathbf{x}}\cap\mathcal{R}_{\mathbf{y}}} \Pr[\mathbf{z}])/(\sum_{\mathbf{z}\in\mathcal{R}_{\mathbf{x}}} \Pr d[\mathbf{z}])$. The *expected coverage* $ec(\mathbf{x})$ for a row $\mathbf{x} = (x_1,\ldots,x_k) \in (\Sigma_\rho \cup \{\star\})^k$ is $\sum_{\mathbf{z}\in\mathcal{R}_{\mathbf{x}}} \Pr[\mathbf{z}|\mathbf{x}] \cdot (\sum_{R\in X_\rho} P_\rho(A_{\mathbf{z},R}))$.

For $1 \leqslant j \leqslant k$, a row $(y_1,\ldots,y_k) \in (\Sigma_\rho \cup \{\star\})^k$ is a *j-successor* of $(x_1,\ldots,x_k)$ if it holds that $x_j = \star$ and $y_j \in \Sigma_\rho$, and that $x_i = y_i$ for $1 \leqslant i \leqslant k$ when $i \neq j$. A row having a $\star$ entry in the jth position has exactly v_ρ j-successors.

Letting $\chi_\rho(R,\mathbf{x})$ be the probability that $P_\rho(A_{\mathbf{z},R}) = \mathbf{true}$ for a randomly chosen completion $\mathbf{z}$ of $\mathbf{x}$, we have $ec(\mathbf{x}) = \sum_{R\in X_\rho} \chi(R,\mathbf{x})$. The selection of a row is accomplished by Select_Average_Row in Algorithm 4, when furnished with a routine Expected_Completions that calculates $\chi_\rho(R,\mathbf{x})$.

Algorithm 4 The Average Algorithm for Hash Families: Selecting a Row

```
SELECT_AVERAGE_ROW(ρ, X_ρ, {P_ρ(A, R)})
    // {P_ρ(A, R)} gives a predicate for each R ∈ X and assignment A to R
    Set r^(0) = {⋆}^k
    for i from 1 to k do
        Choose a coordinate γ for which r_γ^(i−1) = ⋆
        Set maxcov = 0 and choice = ∅
        for σ ∈ Σ_ρ
            Let z be the γ-successor of r^(i−1) with z_γ = σ
            if Pr[z|r^(i−1)] > 0
                Set cov = 0
                for R ∈ X_ρ
                    cov = cov + EXPECTED_COMPLETIONS(ρ, R, z)
                if cov ⩾ maxcov{Set maxcov = cov; choice = z}
        Set r^(i) = choice
    return r^(k)
```

When $\mathbf{r}^{(i-1)}$ has $r_\gamma^{(i-1)} = \star$, and $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(v_\rho)}$ are its γ-successors, $ec(\mathbf{r}^{(i-1)}) = \sum_{i=1}^{v_\rho} \Pr[\mathbf{y}^{(i)}|\mathbf{x}] \cdot ec(\mathbf{y}^{(i)})$. Hence $ec(\mathbf{r}^{(i-1)}) \leqslant ec(\mathbf{r}^{(i)})$ for $1 \leqslant i \leqslant k$. Now $ec(\mathbf{r}^{(0)})$ is the expected number of elements of X covered by a row selected at random from Σ_ρ^k according to the probability distribution. Moreover, $ec(\mathbf{r}^{(k)})$ is the actual number of elements of X covered by the row $\mathbf{r}^{(k)}$, which therefore covers at least the expected number.

It remains to compute $\chi_\rho(R, \mathbf{x})$ by EXPECTED_COMPLETIONS for each row ρ, each requirement R, and an arbitrary $\mathbf{x} \in (\Sigma_\rho \cup \{\star\})^k$. While this can be carried out for various probability distributions, in Algorithm 5 we treat only the case when the probability distribution is uniform.

Algorithm 5 The Average Algorithm: Expected Completions

```
EXPECTED_COMPLETIONS(ρ, R, x)
    // For the uniform distribution
    // R is the set C = {γ_1, ..., γ_t} of columns and the partition C_1, ..., C_s
    Let F = {γ ∈ C: x_γ = ⋆} and F̄ = C \ F
    Set count = 0
    for each assignment A = {a_γi}_1⩽i⩽t with a_γi = x_γi for γ_i ∈ F̄
            and a_γi ∈ Σ_ρ for γ_i ∈ F
        if P_ρ(A, R) = true then count = count + 1
    return count · (v_ρ)^−|F|
```

EXPECTED_COMPLETIONS relies on the fact that, for a completion $\mathbf{z}$ of $\mathbf{x}$, whether $P_\rho(A_{\mathbf{z},R}) = \mathbf{true}$ depends only on the assignment to the coordinates C specified by R. Every completion is equally probable and, once the assignment to coordinates of C is specified, either every completion satisfies the constraint or none does. Therefore we can just treat each assignment to the coordinates of C.

The routines in Algorithms 3, 4, and 5 implement the method of Algorithm 2 for a wide variety of hash families, producing a hash family of size no larger than that produced by applying the average algorithm directly. The improvement is that, by using a method of conditional expectations, the algorithm has running time polynomial in the number of requirements rather than the number of sets, when the strength t and the number of symbols v are fixed. To see this, EXPECTED_COMPLETIONS takes time $O(v^t) = O(1)$. Then when there are r requirements, SELECT_AVERAGE_ROW takes time $O(k \cdot v \cdot r)$, but r is bounded by $(kv)^t$, so the time is $O(k^{t+1})$. When N rows are produced, AVERAGE_HASH_FAMILY takes time $O(N \cdot k^{t+1})$. By Theorem 5.4, N is $O(\log k)$ when t is fixed, and hence the running time is indeed a polynomial in k. A weaker runtime bound, which is still polynomial in k, is obtained when v is not fixed but t remains fixed because $v \leqslant k$ in the problems examined.

7. An illustration – some computational results

Surprisingly, the large constants suppressed in this analysis do not render the method impractical. Table 1 reports some computational results for d-strengthening $\mathsf{DHF}(N; 13, v, t, s)$. Whenever $v' \geqslant v$, a d-strengthening $\mathsf{DHF}(N; 13, v, t, s)$ is also a d-strengthening $\mathsf{DHF}(N; 13, v', t, s)$; we have indicated in italics when the entry is implied in this way.

We outline the method briefly here. For each s-class partition of the t columns, we identify a row pattern by choosing a row that maximizes the number of t-sets of columns separated while using at most d symbols on the t columns, when no separations have already been accomplished. This yields a number of row patterns, one for each partition. We only select rows that have the same multiplicities of symbols as one of the row patterns. Restricting row patterns in this way (equivalently, selecting suitable probabilities for particular rows to be selected) accelerates the method and improves the sizes obtained, sometimes dramatically. Indeed for a 3-strengthening $\mathsf{DHF}(N; 13, 12, 6, 3)$, using this restriction of row selections, the greedy method produced 482 rows; simply selecting rows uniformly at random leads to a bound in our computation of 1969 rows. This underscores the value of selecting rows from a suitable probability distribution.

Table 1
d-strengthening DHF(N; 13, v, t, s).

s	d	t	Number of symbols v										
			2	3	4	5	6	7	8	9	10	11	12
2	2	2	4	3	2	*2*	*2*	*2*	*2*	*2*	*2*	*2*	*2*
	2	3	9	*9*	*9*	*9*	*9*	*9*	*9*	*9*	*9*	*9*	*9*
	2	4	19	*19*	*19*	*19*	*19*	*19*	*19*	*19*	*19*	*19*	*19*
	2	5	50	*50*	*50*	*50*	*50*	*50*	*50*	*50*	*50*	*50*	*50*
	2	6	106	*106*	*106*	*106*	*106*	*106*	*106*	*106*	*106*	*106*	*106*
	3	3		5	4	3	*3*	*3*	*3*	*3*	2	*2*	*2*
	3	4		10	*10*	*10*	*10*	*10*	*10*	*10*	*10*	*10*	*10*
	3	5		24	*24*	*24*	*24*	*24*	*24*	*24*	*24*	*24*	*24*
	3	6		53	*53*	*53*	*53*	*53*	*53*	*53*	*53*	*53*	*53*
	4	4			7	5	*5*	4	*4*	*4*	3	*3*	*3*
	4	5			14	*14*	*14*	*14*	*14*	*14*	*14*	*14*	*14*
	4	6			30	*30*	*30*	*30*	*30*	*30*	*30*	*30*	*30*
	5	5				9	7	6	*6*	5	4	3	*3*
	5	6				18	15	*15*	*15*	*15*	*15*	*15*	*15*
	6	6					13	10	8	6	*6*	4	*4*
3	3	3		7	5	4	3	*3*	*3*	*3*	2	*2*	*2*
	3	4		30	27	*27*	*27*	*27*	*27*	*27*	*27*	*27*	*27*
	3	5		124	*124*	*124*	*124*	*124*	*124*	*124*	*124*	*124*	*124*
	3	6		426	*426*	*426*	*426*	*426*	*426*	*426*	*426*	*426*	*426*
	4	4			13	8	6	5	*5*	4	3	*3*	*3*
	4	5			38	31	*31*	*31*	*31*	*31*	*31*	*31*	*31*
	4	6			117	*117*	*117*	*117*	*117*	*117*	*117*	*117*	*117*
	5	5				20	12	9	7	6	5	3	*3*
	5	6				54	38	33	*33*	*33*	*33*	*33*	*33*
	6	6					27	16	13	10	6	5	4
4	4	4			16	9	6	5	*5*	4	3	*3*	*3*
	4	5			86	63	54	*54*	*54*	*54*	*54*	*54*	*54*
	4	6			484	445	440	*440*	*440*	*440*	*440*	*440*	*440*
	5	5				28	14	9	7	6	5	3	*3*
	5	6				110	69	56	*56*	*56*	*56*	*56*	*56*
	6	6					42	20	13	9	6	5	4
5	5	5				30	14	9	7	6	5	3	*3*
	5	6				200	120	98	*98*	*98*	*98*	*98*	*98*
	6	6					44	20	13	10	6	5	4
6	6	6					45	19	13	9	6	5	4

Simply applying the greedy method results in the earlier rows separating many sets of columns, while later selections separate fewer. Indeed many rows contain entries that have no use in effecting a separation that is needed. Nayeri [48] provides a method for exploiting such 'flexible' positions to repeatedly change entries in the array, eliminating any unnecessary rows that arise in the process. We applied his post-optimization techniques to the results of the greedy method to arrive at the results in Table 1. This can be quite effective: In the case of 3-strengthening DHF(N; 13, 12, 6, 3), the greedy method produced 482 rows, which post-optimization reduced to 426.

We report post-optimized results in Table 1, because results of the greedy methods are particularly well suited to their application. Moreover, in this way, one obtains a clearer picture of the effects of various separation and strengthening conditions, and how they interact.

Because the algorithm selects one row at a time, it is an easy matter to change the permitted number of symbols in the next row, and to change the strengthening requirement for the next row, as the algorithm progresses. Although the analysis of this more general algorithm is unwieldy, its implementation is no more complicated than that of Average_Hash_Family. This generalization was used, for example, to make the hash family in Fig. 4.

8. Conclusion

Column replacement techniques have enjoyed a wide range of applications in testing and measurement problems in which sparsity arises. Compressive sensing has recently been added to the long list of such applications, which already included software interaction testing, combinatorial cryptography, computational learning, combinatorial group testing, and others. The basic column replacement technique was earlier extended to exploit heterogeneity to use many small ingredient arrays rather than one. Despite this, its most severe limitation was that the small ingredients required the same strength

as the array to be produced. By introducing strengthening hash families, we have overcome that limitation here, at least in principle.

In order to demonstrate this, column replacement for compressive sensing has been developed using two standard recovery techniques, to produce measurement matrices for larger strength from ingredients for smaller. This more general column replacement supports recoverability, and has a predictable effect on the RIP parameters that assist with assessing ability to cope with noise. Indeed it also supports efficient recovery schemes using the hierarchical structure of the measurement matrix produced; this substantial topic will be pursued elsewhere by the authors.

In practice, the method developed is only effective when strengthening hash families with 'few' rows can be produced; real applications require that they be produced explicitly and efficiently. When the strength is fixed, two probabilistic methods have been shown to lead to the correct asymptotic bound. One, using the Lovász local lemma, is shown to admit a deterministic, polynomial-time implementation. A second, using the Stein–Lovász–Johnson paradigm, is shown to yield a deterministic, polynomial-time algorithm that is easy to implement, making greedy selections to choose one row at a time, and greedy selections to determine each row one element at a time. In conjunction with a post-optimization technique, this provides a practical method for making strengthening hash families.

Although both algorithms yield the correct asymptotic growth rate for the hash families used, we do not expect either to produce arrays with the minimum number of rows, except in a very few cases. Therefore it remains of interest to develop direct constructions using, for example, error-correcting codes. Strengthening hash families produced in this manner could obviate the need for substantial computation.

Acknowledgements

Thanks to Chris McLean, Peyman Nayeri, and Devon O'Brien, for helpful discussions. This research is supported by ARC DP120103067 (C.J.C., D.H.) and DE120100040 (D.H.).

References

[1] N. Alon, Explicit construction of exponential sized families of k-independent sets, Discrete Math. 58 (1986) 191–193.
[2] R. Baraniuk, Compressive sensing, IEEE Signal Process. Mag. 24 (2007) 227–234.
[3] M. Bazrafshan, Tran van Trung, Bounds for separating hash families, J. Combin. Theory Ser. A 118 (3) (2011) 1129–1135.
[4] R. Berinde, A.C. Gilbert, P. Indyk, H. Karloff, M.J. Strauss, Combining geometry and combinatorics: A unified approach to sparse signal recovery, in: Proc. 46th Annual Allerton Conference on Communication, Control, and Computing, 2008, pp. 798–805.
[5] J. Bierbrauer, H. Schellwat, Almost independent and weakly biased arrays: efficient constructions and cryptologic applications, Lecture Notes in Comput. Sci. 1880 (2000) 533–543.
[6] S.R. Blackburn, T. Etzion, D.R. Stinson, G.M. Zaverucha, A bound on the size of separating hash families, J. Combin. Theory Ser. A 115 (7) (2008) 1246–1256.
[7] R.C. Bryce, C.J. Colbourn, The density algorithm for pairwise interaction testing, Softw. Test. Verif. Reliab. 17 (2007) 159–182.
[8] R.C. Bryce, C.J. Colbourn, A density-based greedy algorithm for higher strength covering arrays, Softw. Test. Verif. Reliab. 19 (2009) 37–53.
[9] E.J. Candès, Compressive sampling, in: Int. Congress of Mathematics, vol. 3, Madrid, Spain, 2006, pp. 1433–1452.
[10] E.J. Candès, The restricted isometry property and its implications for compressed sensing, C. R. Acad. Sci., Ser. 1 Math. 346 (2008) 589–592.
[11] E.J. Candès, J. Romberg, T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inform. Theory 52 (2006) 489–509.
[12] E.J. Candès, T. Tao, Decoding by linear programming, IEEE Trans. Inform. Theory 51 (2005) 4203–4215.
[13] E.J. Candès, T. Tao, Near optimal signal recovery from random projections: Universal encoding strategies, IEEE Trans. Inform. Theory 52 (2006) 5406–5425.
[14] K. Chandrasekaran, N. Goyal, B. Haeupler, Deterministic algorithms for the Lovász local lemma, in: Proc. ACM–SIAM Symposium on Discrete Algorithms (SODA), 2010, pp. 992–1004.
[15] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1) (1998) 33–61.
[16] V. Chvátal, A greedy heuristic for the set-covering problem, Math. Oper. Res. 4 (3) (1979) 233–235.
[17] A. Cohen, W. Dahmen, R.A. DeVore, Compressed sensing and best k-term approximation, J. Amer. Math. Soc. 22 (2009) 211–231.
[18] G. Cohen, S. Litsyn, G. Zémor, On greedy algorithms in coding theory, IEEE Trans. Inform. Theory 42 (1996) 2053–2057.
[19] C.J. Colbourn, Constructing perfect hash families using a greedy algorithm, in: Y. Li, S. Zhang, S. Ling, H. Wang, C. Xing, H. Niederreiter (Eds.), Coding and Cryptology, World Scientific, Singapore, 2008, pp. 109–118.
[20] C.J. Colbourn, Covering arrays and hash families, in: Information Security and Related Combinatorics, in: NATO Peace Inform. Secur. Ser., IOS Press, 2011, pp. 99–136.
[21] C.J. Colbourn, Efficient conditional expectation algorithms for constructing hash families, Lecture Notes in Comput. Sci. 7056 (2011) 144–155.
[22] C.J. Colbourn, D. Horsley, C. McLean, Compressive sensing matrices and hash families, IEEE Trans. Commun. 59 (2011) 1840–1845.
[23] C.J. Colbourn, J. Torres-Jiménez, Heterogeneous hash families and covering arrays, Contemp. Math. 523 (2010) 3–15.
[24] C.J. Colbourn, J. Zhou, Improving two recursive constructions for covering arrays, J. Statist. Theory Practice 6 (2012) 30–47.
[25] G. Cormode, S. Muthukrishnan, Combinatorial algorithms for compressed sensing, Lecture Notes in Comput. Sci. 4056 (2006) 280–294.
[26] Z.J. Czech, G. Havas, B.S. Majewski, Perfect hashing, Theoret. Comput. Sci. 182 (1997) 1–143.
[27] P. Damaschke, Adaptive versus nonadaptive attribute-efficient learning, Mach. Learn. 41 (2000) 197–215.
[28] D. Deng, D.R. Stinson, R. Wei, The Lovász local lemma and its applications to some combinatorial arrays, Des. Codes Cryptogr. 32 (1–3) (2004) 121–134.
[29] R.A. DeVore, Deterministic constructions of compressed sensing matrices, J. Complexity 23 (2007) 918–925.
[30] D.L. Donoho, X. Huo, Uncertainty principles and ideal atomic decomposition, IEEE Trans. Inform. Theory 47 (2001) 2845–2862.
[31] D.-Z. Du, F.K. Hwang, Combinatorial Group Testing and Its Applications, second edition, World Scientific Publishing Co. Inc., River Edge, NJ, 2000.
[32] M. Elad, A.M. Bruckstein, A generalized uncertainty principle and sparse representation in pairs of bases, IEEE Trans. Inform. Theory 48 (2002) 2558–2567.
[33] P. Erdős, L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions, in: Infinite and Finite Sets, vol. II, Colloq., Keszthely, 1973, in: Colloq. Math. Soc. János Bolyai, vol. 10, North-Holland, Amsterdam, 1975, pp. 609–627.

[34] P. Erdős, J.L. Selfridge, On a combinatorial game, J. Combin. Theory Ser. A 14 (1973) 298–301.
[35] J.-J. Fuchs, On sparse representations in arbitrary redundant bases, IEEE Trans. Inform. Theory 50 (2004) 1341–1344.
[36] J.J. Fuchs, Recovery of exact sparse representations in the presence of bounded noise, IEEE Trans. Inform. Theory 51 (2005) 3601–3608.
[37] A.C. Gilbert, M.A. Iwen, M.J. Strauss, Group testing and sparse signal recovery, in: 42nd Asilomar Conference on Signals, Systems, 2008, pp. 1059–1063.
[38] A.C. Gilbert, M.J. Strauss, J. Tropp, R. Vershynin, One sketch for all: Fast algorithms for compressed sensing, in: ACM Symp. on Theory of Computing, 2007, pp. 237–246.
[39] R. Gribonval, M. Nielsen, Sparse representations in unions of bases, IEEE Trans. Inform. Theory 49 (2003) 3320–3325.
[40] M.A. Iwen, Combinatorial sublinear-time Fourier algorithms, Found. Comput. Math. 10 (2010) 303–338.
[41] S. Jafarpour, W. Xu, B. Hassibi, R. Calderbank, Efficient and robust compressed sensing using optimized expander graphs, IEEE Trans. Inform. Theory 55 (2009) 4299–4308.
[42] D.S. Johnson, Approximation algorithms for combinatorial problems, J. Comput. System Sci. 9 (1974) 256–278.
[43] R.M. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations, Plenum, New York, 1972, pp. 85–103.
[44] L. Lovász, On the ratio of optimal integral and fractional covers, Discrete Math. 13 (4) (1975) 383–390.
[45] K. Mehlhorn, Data Structures and Algorithms 1: Sorting and Searching, Springer-Verlag, Berlin, 1984.
[46] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[47] B.K. Natarajan, Sparse approximate solutions to linear systems, SIAM J. Comput. 24 (1995) 227–234.
[48] P. Nayeri, Post-Optimization: Necessity Analysis for Combinatorial Arrays, PhD thesis, Arizona State University, 2011.
[49] C. Nie, H. Leung, A survey of combinatorial testing, ACM Comput. Surv. 43 (2) (2011), #11.
[50] D.J. O'Brien, Exploring hash families and their applications to broadcast encryption, Master's thesis, Arizona State University, 2011.
[51] S.K. Stein, Two combinatorial covering theorems, J. Combin. Theory Ser. A 16 (1974) 391–397.
[52] D.R. Stinson, Tran Van Trung, R. Wei, Secure frameproof codes, key distribution patterns, group testing algorithms and related structures, J. Statist. Plann. Inference 86 (2000) 595–617.
[53] D.R. Stinson, R. Wei, K. Chen, On generalized separating hash families, J. Combin. Theory Ser. A 115 (2008) 105–120.
[54] M. Stojnic, W. Xu, B. Hassibi, Compressed sensing-probabilistic analysis of a null-space characterization, in: Int. Conf. Acoustics, Speech and Signal Processing, 2008, pp. 3377–3380.
[55] J.A. Tropp, Recovery of short, complex linear combinations via l_1 minimization, IEEE Trans. Inform. Theory 51 (2005) 1568–1570.
[56] W. Xu, B. Hassibi, Efficient compressive sensing with deterministic guarantees using expander graphs, in: Proceedings of IEEE Information Theory Workshop, 2007, pp. 414–419.
[57] Y. Zhang, On theory of compressive sensing via ℓ_1-minimization: Simple derivations and extensions, Technical Report CAAM TR08-11, Rice University, 2008.